PSS®SINCAL
Database Interface and Automation

This document describes the organization and the structure of the PSS SINCAL database, shows you how to fill the database with external programs and explains the automation functions of the calculation methods.

1 General Remarks 2

2 Structure of PSS SINCAL Database 4

2.1 PSS SINCAL Networks 4

2.2 Data Model Design Guidelines 5

2.3 Structure of the Database 7

2.4 Database Analysis with the Help of an Example Network 9

2.5 Tables of Network Graphics 18

2.6 Results in the Database 26

3 Filling in the PSS SINCAL Database 29

3.1 Example Program for Filling the Database 29

3.2 Help Program for Creating PSS SINCAL Database 37

4 Automation of the Calculation Methods 42

4.1 Available Automation Functions 48

4.2 Calculation Objects and their Attributes 85

5 Reference 125

5.1 Documentation 125

5.2 PSS SINCAL Architecture 126

5.3 Ready-Made Solutions 126

1 General Remarks

One of the most important characteristics of PSS SINCAL is the complete transparency of the data. With standard methods, you can access input data and calculation results at any time, even if you are not using PSS SINCAL. This transparency is reached by saving all data in a relational database, necessary for the network planning. Unlike other network planning systems that you only fill when you "export", PSS SINCAL uses the database as the central storage medium for all your data.

Calculation methods are relatively easily to use for **individual solutions** because of their open architecture. Normally, you use an existing PSS SINCAL network that you analyze with a "tool" you have created. The "tool" can take advantage of all the functions of the PSS SINCAL calculation methods. You can use any programming or script language you want for your individual solution. The only prerequisite is that you must be able to access COM functions.

If you use a relational database, you can even **couple to a Geographic Information System** (GIS), a Network Information System (NIS) or a network management system. Basically, you distinguish between a pure calculation solution and one that exports all the data to PSS SINCAL. Which solution you choose depends on what you want to achieve.

Pure Calculation Solution

The pure calculation solution is also known as the engine solution. Coupling frequently needed for basic calculation methods (such as, for example, load flow and short circuit) are implemented. Most of the calculations can be started and used from the source system. In this case, maintaining data, presenting and displaying results and coloring are all done in the source system.

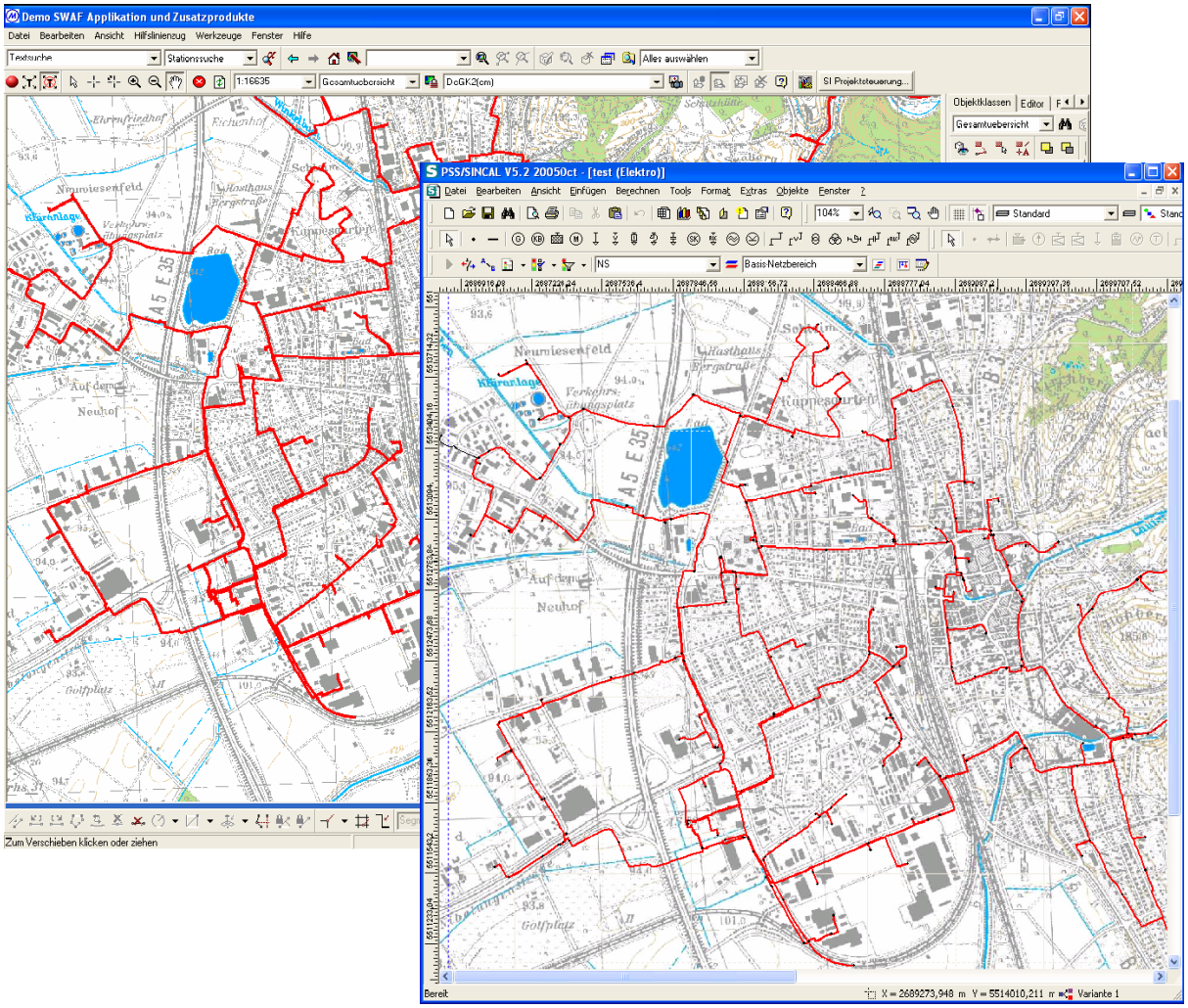
In this solution, only the technical data from the source system are exported to the PSS SINCAL. Finally, automation is used to start the desired PSS SINCAL calculation method. The calculation results are displayed in the source system.

Data Export to PSS SINCAL

The data from the source system are exported to the PSS SINCAL database. Normally, both the technical data for the equipment and the graphic location data are exported.

This solution uses PSS SINCAL to plan and evaluate networks. The complete range of functions of the product can be used.

The following illustration shows Mettenmeier GmbH’s coupling solution (see Reference – Ready-Made Solutions). This is a connection to Smallworld GIS. The illustration shows the same network area in the GIS and in PSS SINCAL.



Generally speaking, this coupling solution is more difficult to implement, since a network diagram has to be generated in addition to the technical data of the equipment. Even determining the appropriate amount of detail is difficult: Detailing is normally much greater in GIS than in a network planning system. A large amount of data might even interfere with the planning. Coupling requires the appropriate amount of detail for processing and planning in PSS SINCAL to be productive.

The advantages of this solution are, however, in particular in its universality. Here – as mentioned above – the full PSS SINCAL range of functions for the planning and evaluation of the networks can be used.

2 Structure of PSS SINCAL Database

This chapter describes the structure of the PSS SINCAL database and explains the data model in detail with the help of a small example network.

2.1 PSS SINCAL Networks

A PSS SINCAL network is made up of a folder pair containing a SIN file and an additional directory.

- {networkname}.sin
- {networkname}_files

The file with extension ".sin" is a PSS SINCAL User Interface help file to simplify network management. Most network-specific settings of the user interface and the supplementary graphics objects of the network are stored in this file. When a network is opened with the Open dialog box, this file is selected.

The directory with the suffix "_files" has all the additional network data. This directory stores the actual network database, the diagram files, various log files and files with the results.

```
Example Ele1.sin
Example Ele1_files
|   database.ini
|   database.mdb
|   database.dia
|
\---NETO
        network.bat
        network.ctl
        ...
```

The file with the ending ".ini " is a configuration file that lets you configure how PSS SINCAL will use the database.

The file with the extension ".mdb" is the network database in Microsoft Access format. All data that describe the network are stored in this file.

If you use a server database system such as ORACLE or SQL Server, for example, there is no MDB file. In this case, the program stores network data directly in the central server database.

Supported Database Systems

Currently following database management systems are supported:

- Microsoft Access (2003, 2007 and 2010)
- Oracle 9i, 10g and 11g
- SQL Server Express 2008 and 2008 R2
- SQL Server 2008 and 2008 R2

2.2 Data Model Design Guidelines

The PSS SINCAL Data Model was developed using the following criteria:

- The PSS SINCAL Data Model is object-oriented.
- All objects are unique and respond to the primary key.
- The standardized layout used through most of the program reduces redundancies and simplifies the search for, and control of, consistency.
- The PSS SINCAL Data Model is separated into categories to simplify evaluations and management.
- There are no limitations as to how the selected data model is implemented in all the PSS SINCAL Relational Database Management systems.

Table Names

Appropriate English terms have been selected as table names. PSS SINCAL uses upper- and lower-case letters to improve legibility.

Essentially, the names of tables differentiate among input data, graphic data, and results.

All tables for **input data** have appropriate simple names:

- Line
- Infeeder
- TwoWindingTransformer
- etc.

All tables with **graphic data** begin with "Graphic":

- GraphicElement
- GraphicLayer
- etc.

The tables for **result data** begin with an abbreviation for the calculation method and end with "Result":

- LFNodeResult
- LFBranchResult
- SC1NodeResult
- SC1BranchResult
- etc.

Keys

Most of the tables in the PSS SINCAL Data Model have **primary keys** to uniquely identify data. The primary key in a table contains the name of the table and has the ending "_ID".

References in the PSS SINCAL Data Model have a **secondary key**. It includes the name of the reference table and has the ending "_ID".

The data type of keys is always "LongInteger".

Examples of primary and secondary keys:

- Table **Element**
  Primary key: Element_ID
  Secondary key: VoltLevel_ID, Variant_ID
- Table **Terminal**
  Primary key: Terminal_ID
  Secondary key: Element_ID, Node_ID

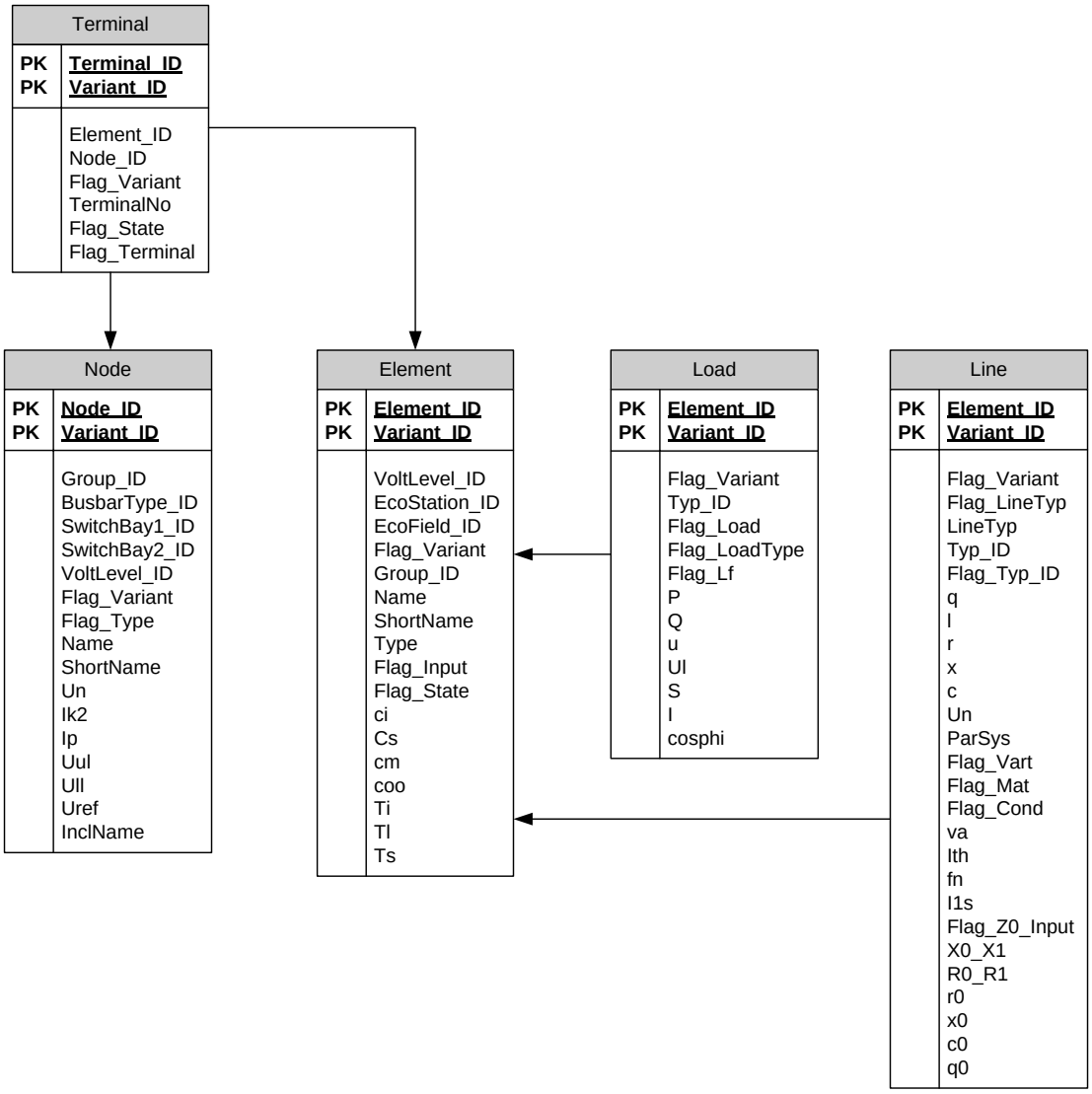
Note: PSS SINCAL uses special algorithms for the management of the key fields. Therefore it is important to ensure that the IDs are generated ascending starting with the smallest possible ID (1). Gaps in the IDs are easily possible, but it should be avoided to store very large numbers, because otherwise problems with PSS SINCAL GUI functions and variant management have to be expected. Also the direct storage of unique GIS IDs in the primary key fields is not allowed. For this purpose, the special **MasterResource** mapping table is available.

Attribute Names

- Whenever possible, attribute names are the same as the corresponding formula sign.
- Attribute names are kept as language-neutral as possible or English terms are used.
- Attribute names are both unique and case-sensitive.
- Numbers replace superscripts and commas (e.g. ' > 1 “ > 2).
- Underscoring ("_") is used as a placement marker for slashes and to connect expressions.
- External and primary keys end with "_ID" (e.g. Element_ID Variant_ID).

2.3 Structure of the Database

The following illustration shows the basic structure of the PSS SINCAL Data Model on the basis of some selected tables.



The **Node** table describes the basic network topology. **Terminal** tables attach node and branch elements to a node and are used to construct the entire network topology.

At the center of the data model, you have the **Element** table. This describes the actual network elements. This table is assigned an additional table for the detailed description of the particular network element:

- Line: Element + Line
- Consumer: Element + Load
- etc.

Input Data Status

The data tables for the elements summarize the data for different calculation methods to keep the data model from becoming too complex. The data table for the Line, for example, contains the data for the load flow calculations, short circuit calculations, harmonic calculations, etc.

All network element attributes are separated into categories. This means that, although data supply summary information in a shared table, data can be selected for a specific calculation method or the present status of the data can be seen.

The table **Element** tables have a **Flag_Input** attribute that stores the current status of the data input for each category. Flags show what data have or have not been entered. Flags for different categories can be created with the help of the bit-wise OR-operator.

Electrical Networks

- Bit 0: Short circuit data
- Bit 1: Load flow data
- Bit 2: Zero-phase sequence data
- Bit 3: Negative-phase sequence data
- Bit 4: Harmonics data
- Bit 5: Dynamics data
- Bit 6: Protection data
- Bit 7: Regulator data
- Bit 8: Reliability data
- Bit 9: Supplementary data
- Bit 10: Measurement data
- Bit 11: Motor start-up data
- Bit 12: Transformer regulator data
- Bit 13: Distance protection data
- Bit 14: Generator unit data
- Bit 15: Transformer unit data
- Bit 16: Direct feeder data for generator
- Bit 17: Equivalent element data
- Bit 39: Dynamic data

Pipe Networks

- Bit 24: Flow data
- Bit 25: Water data
- Bit 26: Gas data
- Bit 27: Heating/Cooling data
- Bit 28: Static data

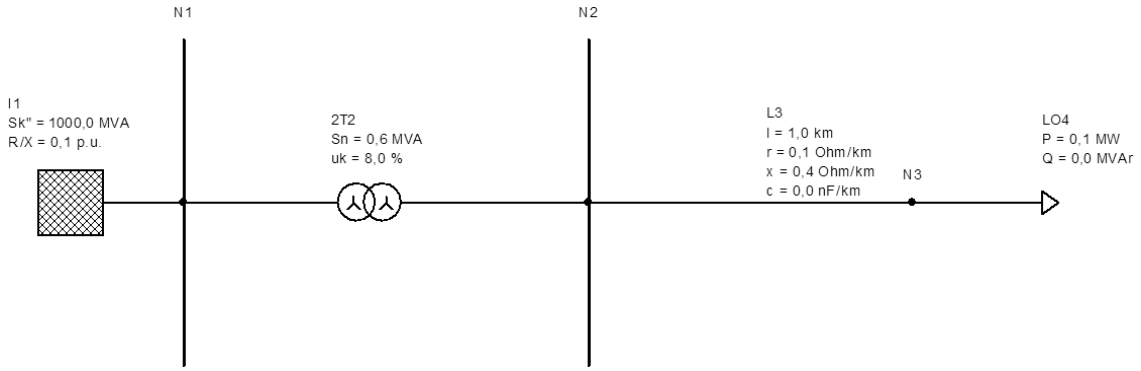
- Bit 30: Equivalent element data

Example for coding short circuit and load flow data for electrical networks:

Bit 0 + Bit 1 = 1 + 2 = 3

2.4 Database Analysis with the Help of an Example Network

The following is an explanation of the data model with the help of a small example network. The PSS SINCAL user interface is used to create a network and analyze the database.

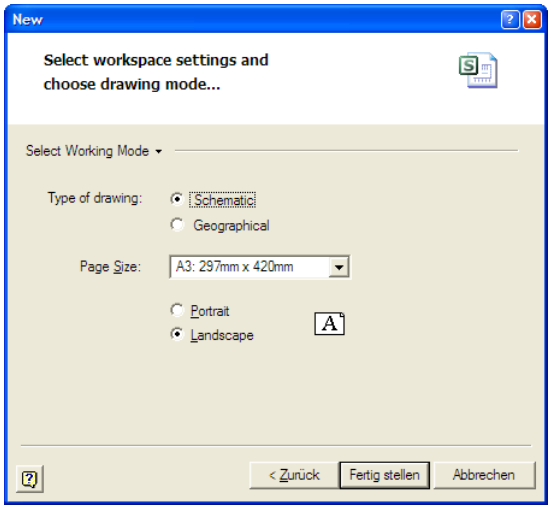


The network displayed here is created step by step. To see how the data model works, the contents of the database are analyzed after each processing step. The following steps are needed to create the example network:

- Step 1: Create a new network
- Step 2: Create network levels
- Step 3: Create busbars
- Step 4: Create the infeeder
- Step 5: Attach the two-winding transformer
- Step 6: Create the line
- Step 7: Attach the consumer

Step 1: Create a New Network

A new schematic electrical network is created in the PSS SINCAL user interface. First **File – New** is clicked in the menu.

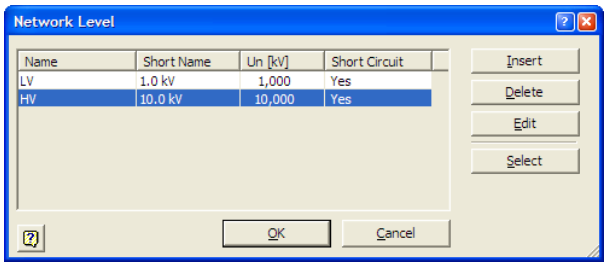


For the example network, the **Schematic** type of drawing has been selected and the page format set to **A3 Landscape**.

Step 2: Create Network Levels

In PSS SINCAL, the network elements need to be assigned to a network level. The network level is used to specify globally valid data for network elements (e.g. rated voltage in electrical networks).

As a default when you generate a new network, PSS SINCAL automatically creates a network level. This is filled with standard values and can be customized. Select **Insert – Network Level** in the menu.



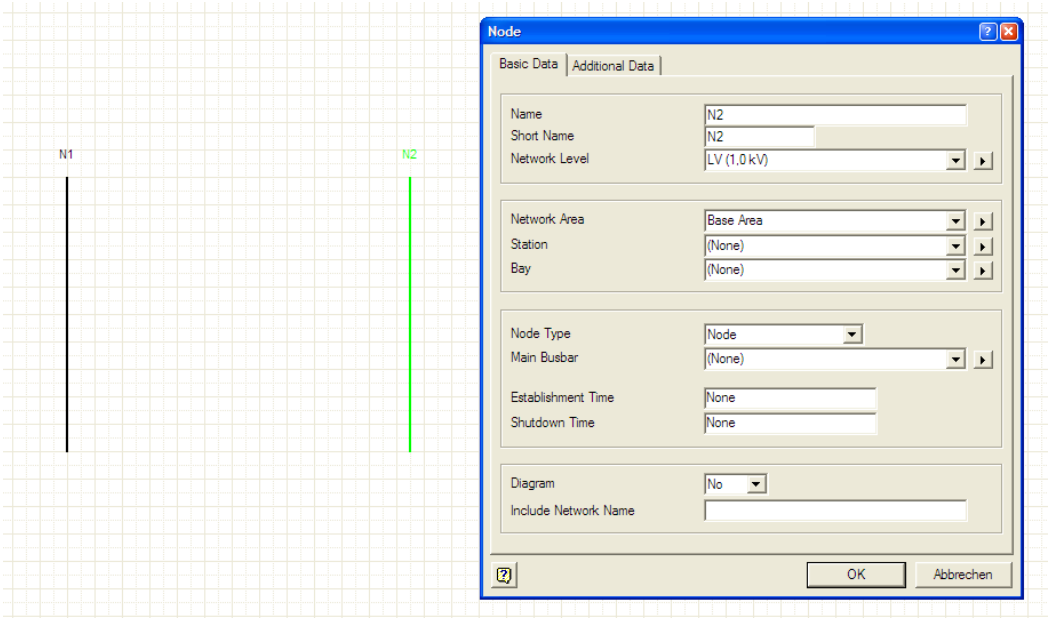
For the example network, the following network levels are created: LV = 1kV , HV = 10kV

Once the network levels have been created, the **VoltageLevel** table contains the following values:

VoltageLevel									
VoltLevel_ID	**Name**	**ShortName**	**Un**	**Uop**	**c**	**cmax**	**…**	**Flag_Variant**	**Variant_ID**
1	LV	1.0 kV	1,0	1,0	1,1	1,1		1	1
2	HV	10.0 kV	10,0	10,0	1,1	1,1		1	1

Step 3: Create Busbars

Now the nodes or busbars can be created. To do so, click **Insert – Node/Busbar – Busbar** in the menu. Finally, both the busbars N1 and N2 are created in the Graphics Editor.



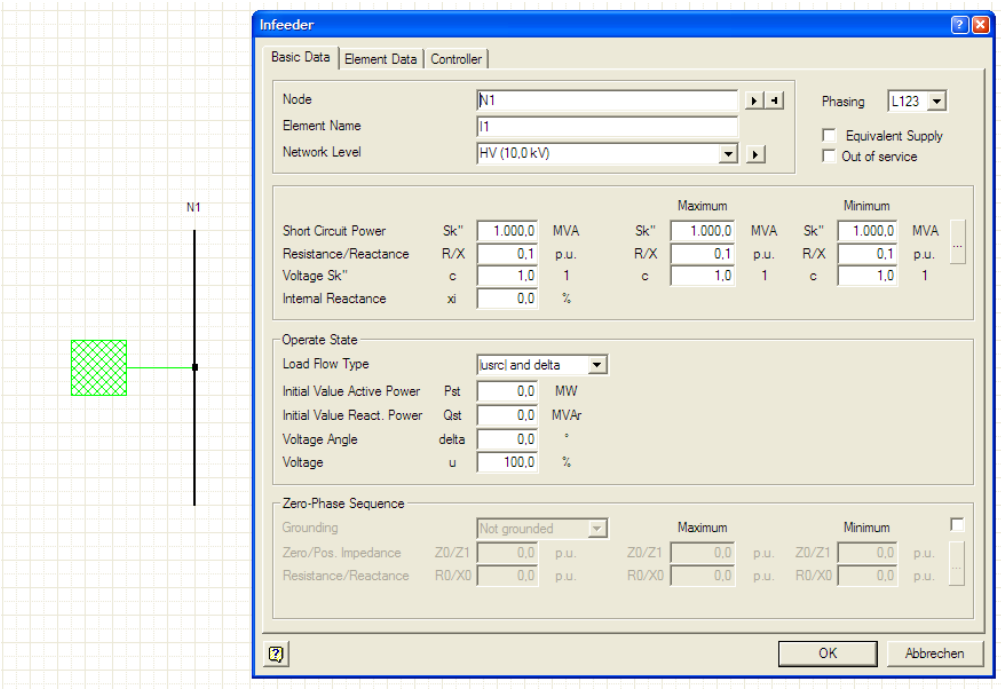
Once both busbars have been created, the **Node** table contains the following values.

Node							
Node_ID	**Group_ID**	**VoltLevel_ID**	**Name**	**Un**	**…**	**Flag_Variant**	**Variant_ID**
1	1	2	N1	10,0		1	1
2	1	1	N2	1,0		1	1

Here you can already see the nodes/busbars (Node) are assigned to the network levels (VoltageLevel) with the secondary key. The busbar N1 has been assigned to the 10 kV network level HV and given the ID 2. The busbar N2 has been assigned to the 1 kV network level LV and given the ID 1.

Step 4: Create the Infeeder

The next step is to create an infeeder. PSS SINCAL has various network elements to simulate infeeders. The following example creates a network infeeder. Click **Insert – Node Element – Infeeder** in the menu to start this process.



The infeeder is a network element with one single terminal. This network element is therefore designated as a node element. In the database, the infeeder is described with the following tables:

- **Element**: This is the basic record for the network element.
- **Infeeder**: This table contains the specific attributes for the infeeder.
- **Terminal**: This table creates the connection between the network element and the nodes/busbars.

Element								
Element_ID	**VoltLevel_ID**	**Group_ID**	**Name**	**Type**	**Flag_Input**	**…**	**Flag_Variant**	**Variant_ID**
1	2	1	I1	Infeeder	3		1	1

The basic data for the network element are stored in the Element table. The record contains the primary key for the network element in the **Element_ID** field. This assures the network element is uniquely identifiable.

The **VoltLevel_ID** field creates a connection to the network level. In this case, this is the 10 kV network level HV: VoltLevel_ID = 2

The **Type** field stores the type of network element. This is an ASCII field and contains exactly the name of the data table for the network element: types = "Infeeder"

The **Flag_Input** field has the code for the network element's input status. The binary value 3 is for the Bit 0 + Bit 1 → short circuit data + load flow data

Infeeder								
Element_ID	**Typ_ID**	**Flag_Typ_ID**	**Flag_Typ**	**Sk2**	**cact**	**R**	**X**	**…**
1	0	0	2	1000,0	1,0	0,0	0,0	

The specific data for network supply are stored in the Infeeder table. The secondary key **Element_ID** creates the connection to the table Element: Element_ID = 1

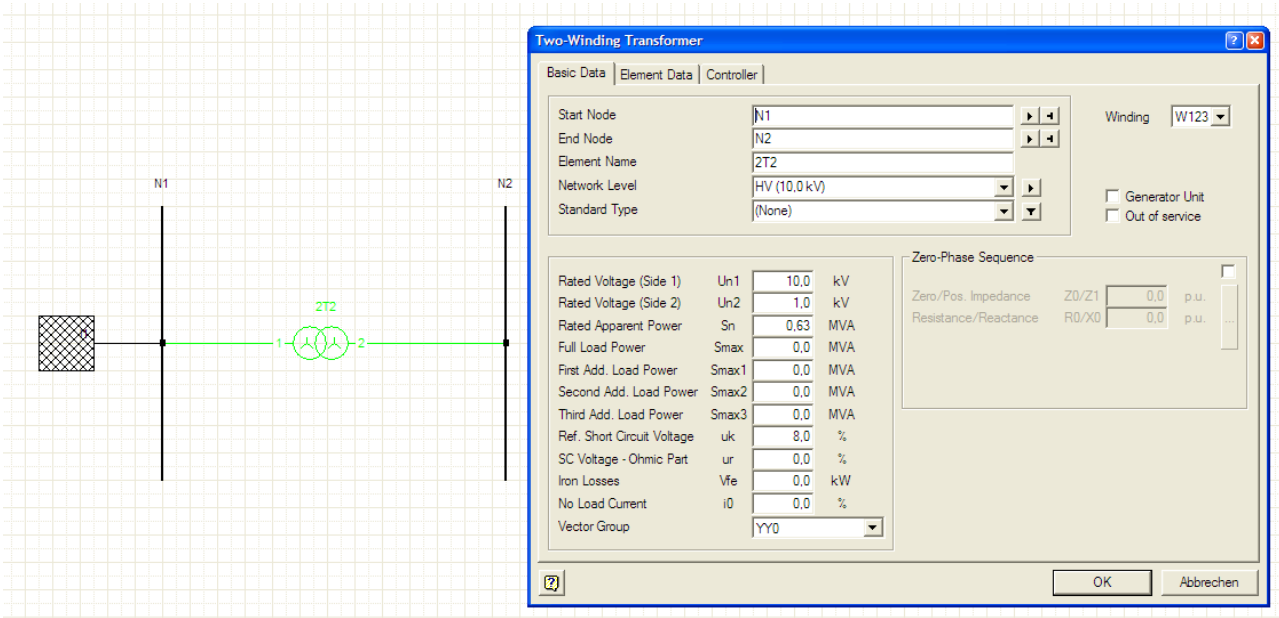
Terminal							
Terminal_ID	**Element_ID**	**Node_ID**	**TerminalNo**	**Flag_State**	**…**	**Flag_Variant**	**Variant_ID**
1	1	1	1	1		1	1

The Terminal table connects the network element to the nodes/busbars. Use the **Element_ID** and **Node_ID** fields for this. In our example, the infeeder I1 is attached to the busbar N1: Element_ID = 1 , Node_ID = 1

The **TerminalNo** field is a counter for the connection number. In network supply (= node element) this is always 1, since this element only has one terminal.

Step 5: Attach the Two-Winding Transformer

Now a two-winding transformer is created between both busbars. Click **Insert – Branch Element – Two-Winding Transformer** in the menu to create the element in the Graphics Editor.



The two-winding transformer is a branch element. This means it has two terminals connecting it to two nodes/busbars. In our example, these are the busbars N1 and N2.

This branch element is described in the database with the following tables:

- **Element**: This is the basic record for the network element.
- **TwoWindingTransformer**: This table contains the specific attributes of the two-winding transformer.
- **Terminal**: This table creates the connection between the network element and the nodes/busbars.

Element								
Element_ID	**VoltLevel_ID**	**Group_ID**	**Name**	**Type**	**Flag_Input**	**…**	**Flag_Variant**	**Variant_ID**
1	2	1	I1	Infeeder	3		1	1
2	2	1	2T2	TwoWinding Transformer	3		1	1

The data in the element table for the two-winding transformer are the same as those previously described for the infeeder.

As with the infeeder, the **Type** field stores the network element: Type = "TwoWindingTransformer"

TwoWindingTransformer							
Element_ID	**Typ_ID**	**Flag_Typ_ID**	**Un1**	**Un2**	**Sn**	**uk**	**…**
2	0	0	10,0	1,0	0,63	8,0	

The input data for the two-winding transformer are stored in the TwoWindingTransformer table. The secondary key **Element_ID** creates the connection to the table element: Element_ID = 2

Terminal							
Terminal_ID	**Element_ID**	**Node_ID**	**TerminalNo**	**Flag_State**	**…**	**Flag_Variant**	**Variant_ID**
1	1	1	1	1		1	1
2	2	1	1	1		1	1
3	2	2	2	1		1	1

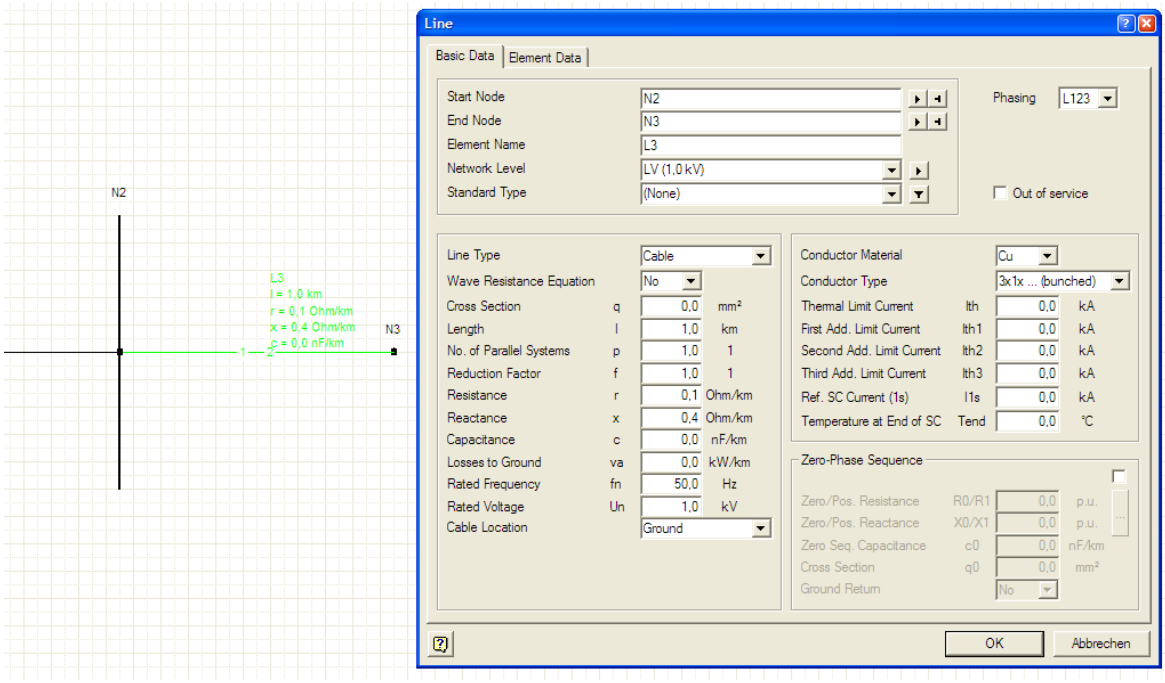
The table Terminal is used to create the connection between the network element and the nodes/busbars. Use the **Element_ID** and **Node_ID** fields for this.

The two-winding transformer is a branch element and thus has two terminals. These attach the transformer 2T2 to the busbars N1 and N2:

- Terminal 1 (Terminal_ID = 2): Element_ID = 2, Node_ID = 1, TerminalNo = 1
- Terminal 2 (Terminal_ID = 3): Element_ID = 2, Node_ID = 2, TerminalNo = 2

Step 6: Create the Line

In the next step, a line is attached to the 1.0 kV busbar. Click **Insert – Branch Element – Line** in the menu to create the element in the Graphics Editor.



The line is a branch element just like the two-winding transformer. This means this is attached to two nodes/busbars. In our example, these are the busbar N2 and the node N3.

This branch element is described in the database with the following tables:

- **Element**: This is the basic record for the network element.
- **Line**: This table contains the specific attributes for the line.
- **Terminal**: This table creates the connection between the network element and the nodes/busbars.

The structure and semantics of the tables are the same as above. For this reason, the tables are only displayed briefly here. So these are easier to recognize, the new records are colored.

Element								
Element_ID	**VoltLevel_ID**	**Group_ID**	**Name**	**Type**	**Flag_Input**	**…**	**Flag_Variant**	**Variant_ID**
1	2	1	I1	Infeeder	3		1	1
2	2	1	2T2	TwoWinding Transformer	3		1	1
3	1	1	L3	Line	3		1	1

Line							
Element_ID	**Typ_ID**	**Flag_Typ_ID**	**q**	**l**	**ParSys**	**Flag_Vart**	**…**
3	0	0	0,0	1,0	1	1	

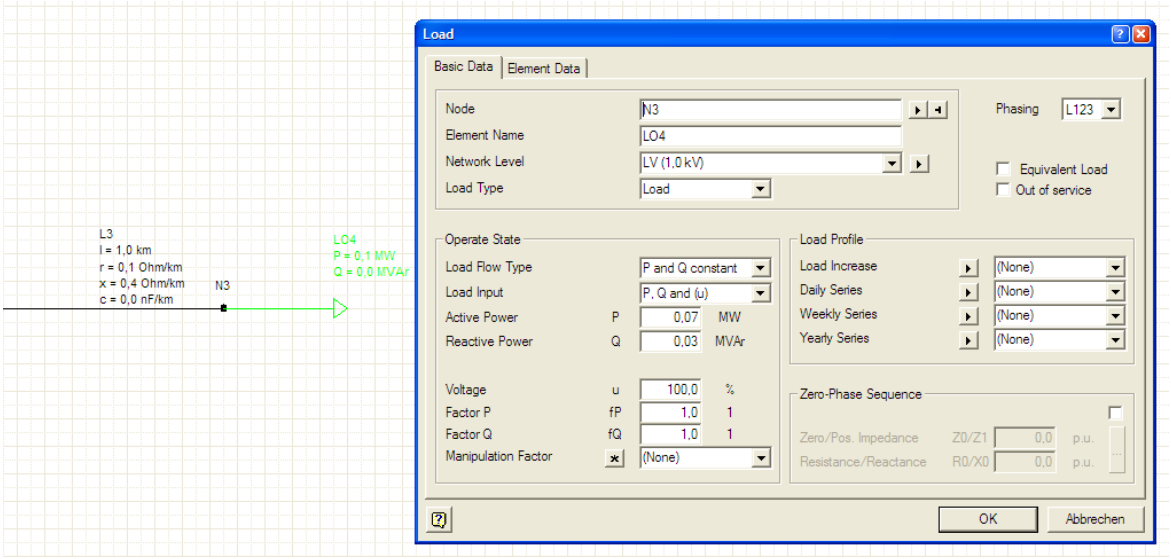
Terminal							
Terminal_ID	**Element_ID**	**Node_ID**	**TerminalNo**	**Flag_State**	**…**	**Flag_Variant**	**Variant_ID**
1	1	1	1	1		1	1
2	2	1	1	1		1	1

3	2	2	2	1		1	1
4	3	2	1	1		1	1
5	3	3	2	1		1	1

Node							
Node_ID	**Group_ID**	**VoltLevel_ID**	**Name**	**Un**	**…**	**Flag_Variant**	**Variant_ID**
1	1	2	N1	10,0		1	1
2	1	1	N2	1,0		1	1
3	1	1	N3	1,0		1	1

Step 7: Attach the Consumer

With attaching a consumer, creating the example network is finished. Click **Insert – Node Element – Load** in the menu to create the element in the Graphics Editor.



The consumer is a node element. In our example, this is attached to the node N3.

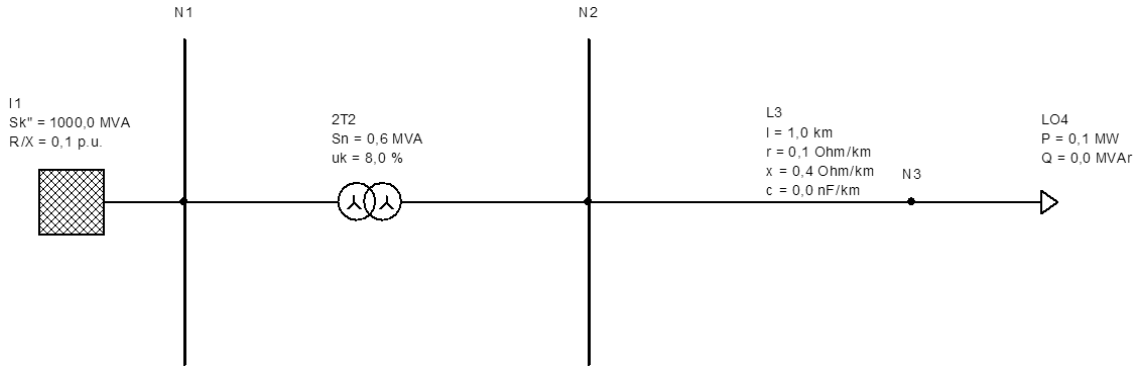
This node element is described in the database with the following tables:

- **Element**: This is the basic record for the network element.
- **Load**: This table contains the specific attributes of the consumer.
- **Terminal**: This table creates the connection between the network element and the nodes/busbars.

The structure and semantics of the tables are the same as above.

2.4.1 Complete Example Network

The example network presented step by step is now complete.



Below the contents of the important tables are displayed briefly to help illustrate the references of the data records once again. To make them easier to recognize, network element records are colored according to their allocation:

- Infeeder: I1
- Two-Winding Transformer: 2T2
- Line: L3
- Consumer: LO4

VoltageLevel									
VoltLevel_ID	**Name**	**ShortName**	**Un**	**Uop**	**c**	**cmax**	**…**	**Flag_Variant**	**Variant_ID**
1	LV	1.0 kV	1,0	1,0	1,1	1,1		1	1
2	HV	10.0 kV	10,0	10,0	1,1	1,1		1	1

Node							
Node_ID	**Group_ID**	**VoltLevel_ID**	**Name**	**Un**	**…**	**Flag_Variant**	**Variant_ID**
1	1	2	N1	10,0		1	1
2	1	1	N2	1,0		1	1
3	1	1	N3	1,0		1	1

Element								
Element_ID	**VoltLevel_ID**	**Group_ID**	**Name**	**Type**	**Flag_Input**	**…**	**Flag_Variant**	**Variant_ID**
1	2	1	I1	Infeeder	3		1	1
2	2	1	2T2	TwoWinding Transformer	3		1	1
3	1	1	L3	Line	3		1	1
4	1	1	LO4	Load	3		1	1

Terminal							
Terminal_ID	**Element_ID**	**Node_ID**	**TerminalNo**	**Flag_State**	**…**	**Flag_Variant**	**Variant_ID**
1	1	1	1	1		1	1

2	2	1	1	1		1	1
3	2	2	2	1		1	1
4	3	2	1	1		1	1
5	3	3	2	1		1	1
6	4	3	1	1		1	1

Infeeder								
Element_ID	**Typ_ID**	**Flag_Typ_ID**	**Flag_Typ**	**Sk2**	**cact**	**R**	**X**	**…**
1	0	0	2	1000,0	1,0	0,0	0,0	

TwoWindingTransformer							
Element_ID	**Typ_ID**	**Flag_Typ_ID**	**Un1**	**Un2**	**Sn**	**uk**	**…**
2	0	0	10,0	1,0	0,63	8,0	

Line							
Element_ID	**Typ_ID**	**Flag_Typ_ID**	**q**	**l**	**ParSys**	**Flag_Vart**	**…**
3	0	0	0,0	1,0	1	1	

Load							
Element_ID	**Flag_Load**	**Flag_LoadType**	**P**	**Q**	**u**	**Ul**	**…**
4	0	0	0,07	0,03	100.0	0.0	

2.5 Tables of Network Graphics

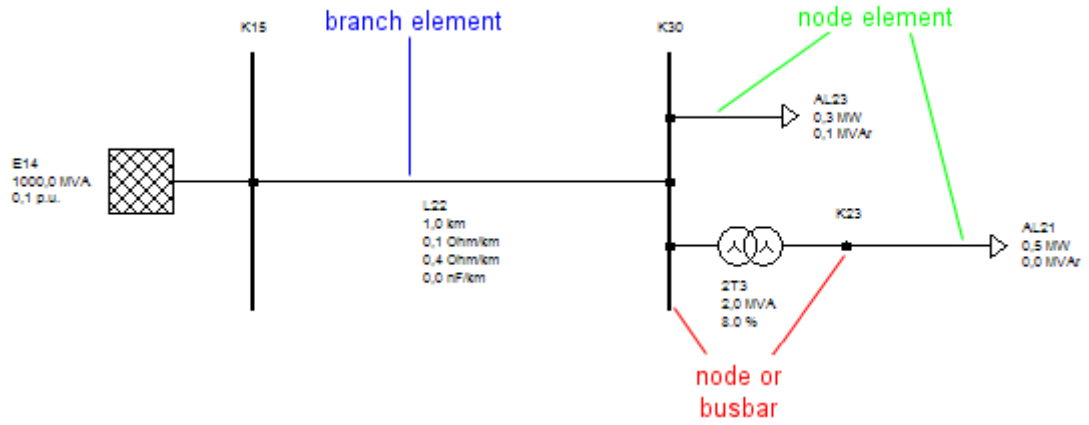
The graphics tables describe the network graphic display. This information is, however, only needed for visualization and processing in the user interface. The graphics tables are not needed for the network calculations. This means that if you are only using PSS SINCAL calculation methods (e.g. for an engines solution), you do not need to fill in the graphics tables.

The following list shows the most important PSS SINCAL graphics tables:

- **Basic tables for graphics elements**
  - GraphicNode: Graphics for nodes/busbars
  - GraphicElement: Graphics for network element symbols
  - GraphicTerminal: Graphics for terminals of network elements
- **Additional tables**
  - GraphicBucklePoint: Buckle points
  - GraphicText: Texts
- **Basic structures**
  - GraphicAreaTile: Area and tile
  - GraphicLayer: Layers
  - GraphicObjectType: Object types

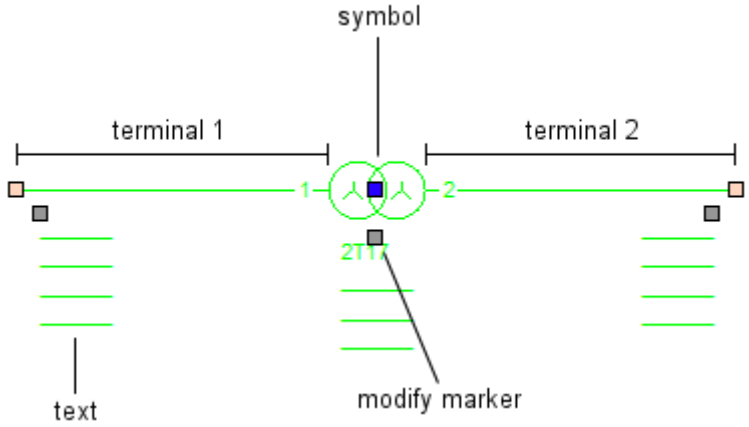
2.5.1 Basic Tables for Graphic Elements

A network's structure is described by its nodes and branches. The branches connect two nodes to each other. A branch (or branch element) goes from the starting node to the end node. Node elements are connected to the nodes.



The simple network elements are **nodes** and **busbars**. They have only one symbol and text. All node elements and branch elements are connected to these.

Network elements are more complex graphics elements. They are composed of symbols, terminals and text.



In our example, a branch element – a two-winding transformer – is selected. As can be seen in this example, components of the network elements are easy to recognize: the displayed branch element consists of a symbol, both terminals (linking the symbol to the node) and texts.

Node elements consist of one terminal, a symbol and text.

Branch elements consist of two terminals, a symbol and text.

A special form of branch elements is the three-winding transformer. Unlike all other branch elements, it connects to three, rather than two, nodes or busbars.

GraphicNode (Graphics for Nodes/Busbars)

This table describes the graphics attributes for nodes/busbars.

Attribute name	Data type	Unit	Description
GraphicNode_ID	Long Integer		Primary Key – Graphic Node
GraphicLayer_ID	Long Integer		Secondary Key – Layer
GraphicType_ID	Long Integer		Secondary Key – Object Type
GraphicText_ID1	Long Integer		Secondary Key – Text 1
GraphicText_ID2	Long Integer		Secondary Key – Text 2
Node_ID	Long Integer		Secondary Key – Node
FrgndColor	Long Integer	RGB	Line Color
BkgndColor	Long Integer	RGB	Background Color
PenStyle	Integer		Pen Style 0: Straight line 1: Small dotted 2: Dotted 3: Straight line – point – straight line 4: Straight line – point – point – straight line
PenWidth	Integer	0.25mm	Pen Width
NodeSize	Integer	0.25mm	Symbol Size Factor
NodeStartX	Double	m	Node Start X-Coordinate
NodeStartY	Double	m	Node Start Y-Coordinate
NodeEndX	Double	m	Node End X-Coordinate
NodeEndY	Double	m	Node End Y-Coordinate
SymType	Integer		Node Symbol Type 0: No symbol 1: Circle 2: Rectangle 3: Busbar
Flag	Long Integer		Flag
Variant_ID	Long Integer		Secondary Key – Variant
Flag_Variant	Integer		Element of Current Variant 0: No 1: Yes
GraphicArea_ID	Long Integer		Secondary Key – Graphic Area/Tile

The secondary key **GraphicText_ID1** assigns a graphic text object. This means an individual text object will be displayed in the Graphics Editor with its own position and graphics attributes. If you wish, you can initialize the field with "NULL". Then PSS SINCAL will display the text with default attributes in the Graphics Editor, but the text cannot be edited manually.

The secondary key **GraphicText_ID2** is not implemented at this time and should therefore always be initialized with "NULL".

GraphicElement (Graphics for Network Element Symbols)

This table describes the graphics attributes for network element symbols.

Attribute name	Data type	Unit	Description
GraphicElement_ID	Long Integer		Primary Key – Graphic Element
GraphicLayer_ID	Long Integer		Secondary Key – Layer
GraphicType_ID	Long Integer		Secondary Key – Object Type

GraphicText_ID1	Long Integer		Secondary Key – Graphic Text 1
GraphicText_ID2	Long Integer		Secondary Key – Graphic Text 2
Element_ID	Long Integer		Secondary Key – Element
SymbolDef	Long Integer		Set Symbol Properties as Default
FrgndColor	Long Integer	RGB	Line Color
BkgndColor	Long Integer	RGB	Background Color
PenStyle	Integer		Pen Style 0: Straight line 1: Small dotted 2: Dotted 3: Straight line – point – straight line 4: Straight line – point – point – straight line
PenWidth	Integer	0.25mm	Pen Width
SymbolSize	Integer	0.25mm	Symbol Size Factor
SymCenterX	Double	m	Symbol Center X-Coordinate
SymCenterY	Double	m	Symbol Center Y-Coordinate
SymbolType	Long Integer		**Symbol Type Electrical Networks:** 9: Synchronous machine 10: Power unit 11: Infeeder 12: Asynchronous machine 13: Load 15: Shunt impedance 16: Shunt reactor 17: Shunt capacitor 18: Static compensator 19: Line 20: Two-winding transformer 21: Three-winding transformer 22: Serial reactor 23: Serial capacitor 24: Shunt ripple control transmitter 25: Serial ripple control transmitter 26: Shunt RLC-circuit 27: Serial RLC-circuit 29: Harmonic resonance network 193: DC-Infeeder 194: DC-Serial element 123: Variable Shunt Element 124: Variable Serial Element **Symbol Type Pipe Networks:** 9: Water Tower 10: Infeeder Pump 11: Infeeder Gas 12: Infeeder Heating/Cooling 13: Consumer 14: Pressure Buffer 15: Leakage 16: Temperature Regulator 17: Line 18: Pressure Increase Pump 19: Const. Pressure Decrease/Const. Flow 20: Pressure Regulator 21: Compressor 22: Heat Exchanger 23: Sliding Valve/Non-Return Valve
SymbolNo	Integer		Symbol Number
Flag	Long Integer		Flag
Variant_ID	Long Integer		Secondary Key – Variant
Flag_Variant	Integer		Element of Current Variant 0: No 1: Yes
GraphicArea_ID	Long Integer		Secondary Key – Graphic Area/Tile

The **SymbolType** field is particularly important here. It needs to be properly initialized or the graphic data will not be assigned – or they will be assigned incorrectly – to network element data in the PSS SINCAL user interface.

The **SymbolDef** field is used to enhance the control of the network element symbols. For coupling solutions this should be initialized with "-1".

The secondary key **GraphicText_ID1** assigns a graphic text object. This means an individual text object will be displayed in the Graphics Editor with its own position and graphics attributes. If you wish, you can initialize the field with "NULL". Then PSS SINCAL will display the text with default attributes in the Graphics Editor, but the text cannot be edited manually.

The secondary key **GraphicText_ID2** is not implemented at this time and should therefore always be initialized with "NULL".

GraphicTerminal (Graphics for Terminals of Network Elements)

This table describes the graphics attributes for network element terminals.

Attribute name	Data type	Unit	Description
GraphicTerminal_ID	Long Integer		Primary Key – Graphic Terminal
GraphicElement_ID	Long Integer		Secondary Key – Graphic Element
GraphicText_ID	Long Integer		Secondary Key – Graphic Text
Terminal_ID	Long Integer		Secondary Key – Terminal
PosX	Double	m	X-Coordinate
PosY	Double	m	Y-Coordinate
FrgndColor	Long Integer	RGB	Line Color
PenStyle	Integer		Pen Style 0: Straight line 1: Small dotted 2: Dotted 3: Straight line – point – straight line 4: Straight line – point – point – straight line
PenWidth	Integer	0.25mm	Pen Width
SwtType	Integer		Switch Type 0: No Type 1: Type 1 2: Type 2 3: Type 3 4: Type 4 5: Type 5 6: Type 6
SwtAlign	Integer		Switch Direction 0: Automatic 1: Position 1 2: Position 2 3: Position 3 4: Position 4
SwtNodePos	Double	0.25mm	Switch Distance to Node
SwtFactor	Integer	0.25mm	Switch Size Factor
SwtFrgndColor	Long Integer	RGB	Switch Line Color
SwtPenStyle	Integer		Switch Pen Style 0: Straight line 1: Small dotted 2: Dotted 3: Straight line – point – straight line 4: Straight line – point – point – straight line

SwtPenWidth	Integer	0.25mm	Switch Pen Width
SymbolType	Integer		Symbol Type
SymbolAlign	Integer		Symbol Direction
SymbolNodePos	Double	0.25mm	Symbol Distance to Node
SymbolFactor	Integer		Symbol Size Factor
SymbolFrgndColor	Long Integer	RGB	Symbol Line Color
SymbolPenStyle	Integer		Symbol Pen Style
SymbolPenWidth	Integer	0.25mm	Symbol Pen Width
TextAlign	Integer		Adjust Text
Flag	Long Integer		Flag
Variant_ID	Long Integer		Secondary Key – Variant
Flag_Variant	Integer		Element of Current Variant 0: No 1: Yes
GraphicArea_ID	Long Integer		Secondary Key – Graphic Area/Tile

The fields **Pos_X** and **Pos_Y** define the connection point between the terminal and the node or the busbar. With a node, this is always the center. With a busbar, this can be anywhere on the busbar.

The secondary key **GraphicText_ID** assigns a graphic text object. This means an individual text object will be displayed in the Graphics Editor with its own position and graphics attributes. If you wish, you can initialize the field with "NULL". Then PSS SINCAL will display the text with default attributes in the Graphics Editor, but the text cannot be edited manually.

2.5.2 Additional Tables

GraphicText (Text Objects)

This table specifies individual text objects for nodes/busbars and network elements.

Attribute name	Data type	Unit	Description
GraphicText_ID	Long Integer		Primary Key – Graphic Text
GraphicLayer_ID	Long Integer		Secondary Key – Layer
Font	Text (20)		Font
FontStyle	Integer		Style 16: Standard 17: Fat 18: Cursive
FontSize	Integer		Text Height
TextAlign	Integer		Text Alignment 0: Left 1: Middle 2: Right
TextOrient	Integer	0,1 °	Text Orientation
TextColor	Long Integer	RGB	Text Color
Visible	Integer		Visible 0: No 1: Yes
AdjustAngle	Integer		Adjust Text Angle 0: None 1: Horizontal or vertical 2: Direction of element
Angle	Double	°	Text Angle

Pos1	Double	m	Distance X-Direction
Pos2	Double	m	Distance Y-Direction
Flag	Long Integer		Flag
RowTextNo	Integer	1	Number of Rows
AngleTermNo	Integer	1	Partial Terminal Align Number
Variant_ID	Long Integer		Secondary Key – Variant
Flag_Variant	Integer		Element of Current Variant 0: No 1: Yes

Caution: A text object may only be used once. You cannot use the same text object for different elements!

GraphicBucklePoint (Bends for Terminals)

This table specifies bends for network element terminals.

Attribute name	Data type	Unit	Description
GraphicPoint_ID	Long Integer		Primary Key – Buckle Point
GraphicTerminal_ID	Long Integer		Secondary Key – Graphic Terminal
NoPoint	Integer	1	Buckle Point Number
PosX	Double	m	Buckle Point X-Coordinate
PosY	Double	m	Buckle Point Y-Coordinate
Variant_ID	Long Integer		Secondary Key – Variant
Flag_Variant	Integer		Element of Current Variant 0: No 1: Yes

The fields **PosX** and **PosY** define the graphic position of the bend.

The **NoPoint** field determines the sequence of the bends. The bends are numbered sequentially from the network element's symbol point to the connection point at the node/busbar.

2.5.3 Basic Structures

GraphicAreaTile (Worksheet Settings)

This table describes the worksheet. Basically, both the page size and the scale are defined.

Attribute name	Data type	Unit	Description
GraphicArea_ID	Long Integer		Primary Key – Area/Tile
AreaWidth	Double	cm	Page Width
AreaHeight	Double	cm	Page Height
VectorX	Double	m	Zero-Coordinate Placement X
VectorY	Double	m	Zero-Coordinate Placement Y
GridWidth	Integer	0.25mm	Grid Spacing Width
GridHeight	Integer	0.25mm	Grid Spacing Height

Scale1	Integer		Predefined Scale 0: 1:100000000 1: 1:10000000 2: 1:1000000 3: 1:100000 4: 1:10000 5: 1:1000 6: 1:100 7: 1:10 8: 1:1
Scale2	Integer		Display Unit 0: mm 1: cm 2: m 3: km 4: Inch 5: Feet 6: Yards 7: Miles
Flag	Long Integer		Network Working Mode 1: Geographical 2: Schematic
Variant_ID	Long Integer		Secondary Key – Variant
ScalePaper	Double		Scale Paper
ScaleReal	Double		Scale Real
Name	Text (50)		Name of View
TileIndex	Text (8)		Index of Tile

In a PSS SINCAL network, more than one worksheet can be created. Simply create multiple records in this table. In all graphics tables, the **GraphicArea_ID** is available as a secondary key. This specifies which worksheet the respective graphic is assigned to.

GraphicLayer (Graphic Layer)

This table specifies a graphic layer. All screen elements are assigned to a graphic layer. The graphic layer lets you control which network elements are displayed on the screen.

Attribute name	Data type	Unit	Description
GraphicLayer_ID	Long Integer		Primary Key – Layer
Name	Text (50)		Layer Name
Visible	Integer		Visible 0: Not visible 1: Visible on screen 2: Visible on print 3: Visible on screen and print
Locked	Integer		Locked for Working 0: No 1: Yes
Type	Integer		Plot Header 0: No 1: Yes
Flag	Long Integer		Order Flag
Variant_ID	Long Integer		Secondary Key – Variant
Flag_Variant	Integer		Element of Current Variant 0: No 1: Yes
VisibleZF	Integer		Visible at Zoom Factor
GraphicArea_ID	Long Integer		Secondary Key – Graphic Area/Tile

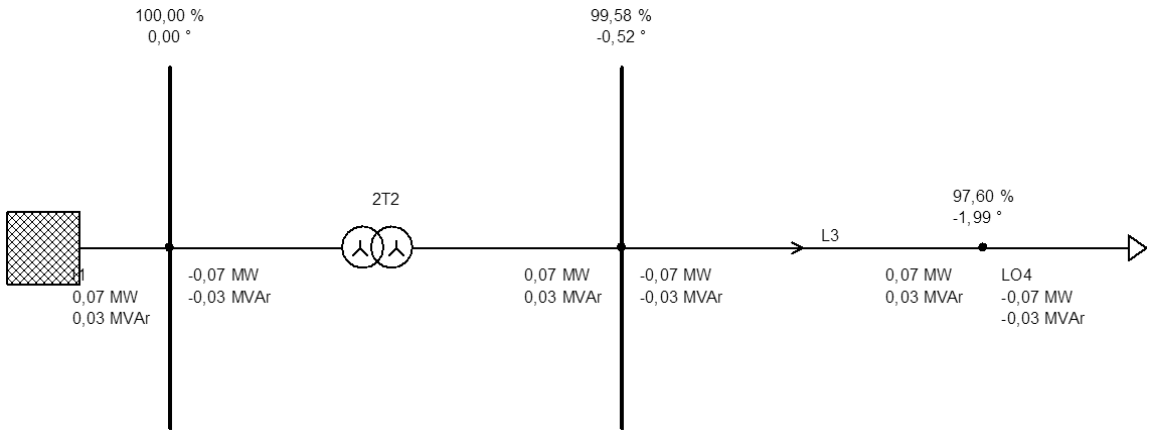
GraphicObjectType (Object Type)

An object type is assigned to all graphic network elements. This object type can help control the legend scope in the network diagram.

Attribute name	Data type	Unit	Description
GraphicType_ID	Long Integer		Primary Key – Object Type
ParentType_ID	Long Integer		Secondary Key – Higher-Level Object Type
Name	Text (50)		Object Type Name
Visible	Integer		Visible Object Type 0: No 1: Yes
Locked	Integer		Locked Object Type
Type	Integer		Object Type
Flag	Long Integer		Flag (not in use)
Variant_ID	Long Integer		Secondary Key – Variant
Flag_Variant	Integer		Element of Current Variant 0: No 1: Yes
VisibleZF	Integer		Visible at Zoom Factor

2.6 Results in the Database

PSS SINCAL stores calculation results in the database in the same way as input data. This is done automatically, once a calculation has been performed successfully. The results in the database can be accessed at any time by your own applications.



The most important tables showing the results for electrical networks are:

- **Load flow**
  - LFNodeResult: Load flow node results
  - LFBranchResult: Load flow branch results
  - ULFNodeResult: Unbalanced load flow node results
  - ULFBranchResult: Unbalanced load flow branch results
  - LFGroupResult: Load flow area results
  - LFParNetLossesResult: Load flow subnetwork losses results

- LFAccurResult: Load flow accuracy results

- **Short circuit**
  - SC3NodeResult: Node results – 3-phase short circuit
  - SC3BranchResult: Branch results – 3-phase short circuit
  - SC1NodeResult: Node results – 1-phase short circuit
  - SC1BranchResult: Branch results – 1-phase ground fault
- **Optimizations**
  - SeparationResult: Separation results
  - InstallCompResult: Compensation power results
- **Harmonics and ripple control**
  - HarBranchResult: Harmonics branch results
  - HarNodeResult: Harmonics node results
  - RCBranchResult: Branch results – ripple control
  - RCNodeResult: Node results – ripple control
  - RCTransmitterResult: Transmitter results – ripple control
- **Reliability**
  - RelResult: Reliability node results
  - RelNetResult: Reliability network results
  - RelGroupResult: Reliability group results

The basic structure of the result tables is displayed according to load flow results. For a detailed description of all available result tables with their attributes, see the **Database Description Manual**.

LFNodeResult (Load Flow Node Results)

This table contains node results from load flow calculations. Node results are assigned by the secondary key **Node_ID**.

Attribute name	Data type	Unit	Description
Result_ID	Long Integer		Primary Key – Result
Node_ID	Long Integer		Secondary Key – Node
Variant_ID	Long Integer		Secondary Key – Variant
U	Double	kV	Node Voltage
U_Un	Double	%	Node Voltage/Rated Node Voltage
phi	Double	°	Angle – Slack Voltage
P	Double	MW	Active Power
Q	Double	Mvar	Reactive Power
S	Double	MVA	Apparent Power
t	Integer	h	Key for Time
tdiag	Double	s	Time – Direct Diagram Connection (22:00 = 22 * 3600)
Flag_Result	Integer		Result Type 0: Load flow 1: Load profile 2: Load development
ResDate			Date
ResTime	Double	h	Time

Flag_State	Integer		State 1: Ok 2: Limit reached
Loading	Double	1	Factor due to Extended Calculation
Uph	Double	kV	Phase Node Voltage
Uph_Unph	Double	%	Node Phase Voltage/Rated Node Phase Voltage
phi_ph	Double	°	Angle – Slack Phase Voltage
U_Uref	Double	%	Voltage/Reference Voltage
Uph_Urefph	Double	%	Phase Voltage/Reference Voltage

LFBranchResult (Load Flow Branch Results)

This table contains branch results from load flow calculations. The results are provided for individual terminals. Results for network elements are assigned by the secondary key **Terminal1_ID**.

Attribute name	Data type	Unit	Description
Result_ID	Long Integer		Primary Key – Result
Terminal1_ID	Long Integer		Secondary Key – Connection
Terminal2_ID	Long Integer		Secondary Key – Neighbor Terminal
Variant_ID	Long Integer		Secondary Key – Variant
P	Double	MW	Active Power
Q	Double	Mvar	Reactive Power
S	Double	MVA	Apparent Power
cos_phi	Double	pu	Power Factor
I	Double	kA	Current
Inb	Double	%	Basic Rating
Pl	Double	MW	Active Power – Losses
Ql	Double	Mvar	Reactive Power – Losses
Sl	Double	MVA	Apparent Power Losses
dU	Double	kV	Series Voltage Drop
deltaphi	Double	°	Phase Rotation
Sn	Double	MVA	Basic Apparent Power
S_Sn	Double	%	Apparent Power/Basic Apparent Power
Inp	Double	kA	Basic Current – Side 1 (primary)
I_Inp	Double	%	Current/Basic Current – Side 1(primary)
Ins	Double	kA	Basic Current – Side 2 (secondary)
I_Ins	Double	%	Current/Basic Current – Side 2 (secondary)
t	Integer	h	Key for Time
tdiag	Double	s	Time – Direct Diagram Connection (22:00 = 22 * 3600)
Flag_Result	Integer		Result Type 0: Load flow 1: Load profile 2: Load development
ResDate			Date
ResTime	Double	h	Time
Flag_State	Integer		State 1: Ok 2: Limit reached
Inb1	Double	%	First Additional Rating
Inb2	Double	%	Second Additional Rating
Inb3	Double	%	Third Additional Rating

3 Filling in the PSS SINCAL Database

This chapter explains how to fill the PSS SINCAL database manually with your own applications.

3.1 Example Program for Filling the Database

When you install PSS SINCAL, there is an example program that teaches you how to fill the database. The example program has been written in VBS (Visual Basic Script). This can be executed with the standard Windows Scripting Host and without any additional software on all current Windows platforms.

The "ImportDB.vbs" example program is in the PSS SINCAL "Batch" directory.

Simply enter the example program at the prompt to start. If you do not add any additional parameters, PSS SINCAL will display hints on how to use this:

```
>cscript.exe ImportDB.vbs
Usage: cscript.exe ImportDB.vbs ImportDB.mdb SincalDB.mdb MODE

MODE: E ... Import data for Electricity
MODE: W ... Import data for Water

This program reads data from ImportDB and writes the data into the SincalDB.
```

To import data, you need a prepared database with import data **ImportDB.mdb** and the appropriate PSS SINCAL network database **SincalDB.mdb**. The **MODE** parameter differentiates between electrical networks and pipe networks.

Start "ImportDB.vbs" with the correct parameters:

```
>cscript.exe ImportDB.vbs EleData.mdb EleTest.mdb E
Init IDs...
Reading Nodes...
Reading Lines...
Reading Loads...
Reading Transformers...
Writing Data...
Node: 27/27
Element: 48/48
Terminal: 82/82
Line: 32/32
TwoWindingTransformer: 2/2
Load: 14/14
GraphicNode: 27/27
GraphicTerminal: 82/82
GraphicElement: 48/48

Successfully finished import to D:\Network\_Import\GIS\EleTest.mdb.

Inserted 362 records in 0.01 seconds.
```

3.1.1 How the Example Program Works

The basic sequence of functions in the example program is relatively simple:

- First common initializations are performed.
- Then data from the sources database are read out and converted to appropriate data structures.
- Finally, data are written to the PSS SINCAL network database using SQL instructions.

In the source text, this looks as follows:

```
' Execute the selected option
Select Case strParam
    Case "E"     bElectro = True
                 Call InitIDs()
                 Call ReadNodes( 1 )
                 Call ReadLines( 1 )
                 Call ReadLoads( 1 )
                 Call ReadTransformers( 1 )
    Case "W"     bElectro = False
                 Call InitIDs()
                 Call ReadFlowNodes( 1 )
                 Call ReadFlowLines( 1 )
    Case Else    Call Usage()
End Select

If ErrorCheck( "Error while reading input data!" ) Then WScript.Quit

' Write data from arrays to SINCAL database
Call WriteSINCAL()
If ErrorCheck( "Error while writing data!" ) Then WScript.Quit
```

Perform Initializations – InitIDs

InitIDs determines the initial values for the primary keys. The corresponding PSS SINCAL network database tables are opened and the maximum value for the primary key is determined. These values are then stored as global variables.

```
'-------------------------------------------------------------------------
' Init startIDs for DB & Init base table names
'-------------------------------------------------------------------------
Sub InitIDs

    WScript.Echo "Init IDs..."

    Call OpenDatabase( strSINCALdb )

    If bElectro = True Then
        strTableNode            = "Node"
        strTableElement         = "Element"
        strTableTerminal        = "Terminal"
        strTableGraphicText     = "GraphicText"
        strTableGraphicNode     = "GraphicNode"
        strTableGraphicElement  = "GraphicElement"
        strTableGraphicTerminal = "GraphicTerminal"
    Else
        strTableNode            = "FlowNode"
        strTableElement         = "FlowElement"
        strTableTerminal        = "FlowTerminal"
        strTableGraphicText     = "FlowGraphicText"
        strTableGraphicNode     = "FlowGraphicNode"
        strTableGraphicElement  = "FlowGraphicElement"
        strTableGraphicTerminal = "FlowGraphicTerminal"
    End If
```

```
    iNodeID             = 1 + ReadMaxID( strTableNode, "Node_ID" )
    iElementID          = 1 + ReadMaxID( strTableElement, "Element ID" )
    iTerminalID         = 1 + ReadMaxID( strTableTerminal, "Terminal ID" )

    iGraphicTextID      = 1 + ReadMaxID( strTableGraphicText, "GraphicText ID" )
    iGraphicNodeID      = 1 + ReadMaxID( strTableGraphicNode, "GraphicNode_ID" )
    iGraphicElementID   = 1 + ReadMaxID( strTableGraphicElement, "GraphicElement_ID" )
    iGraphicTerminalID  = 1 + ReadMaxID( strTableGraphicTerminal, "GraphicTerminal ID" )

    Call CloseDatabase()

End Sub
```

Read out Node Data from the Database – ReadNodes & AddNode

ReadNodes reads the node data from the sources database and converts this to PSS SINCAL format. When the data are converted, they generate the proper SQL command at the same time.

```
'-------------------------------------------------------------------------------
' Read Node data from IMPORT DB
'-------------------------------------------------------------------------------
Sub ReadNodes( iMode ) ' iMode = 1 ... Normal Mode, 0 ... Only init
    Dim rsNode

    If iMode = 0 And iCntNode > 0 then Exit Sub

    WScript.Echo "Reading Nodes..."

    Call OpenDatabase(strIMPORTdb)
    Call OpenRecordset( "SELECT Name AS ID, Name, ShortName, NodeType, NetworkLevel, …
                      & "FROM Node", rsNode )

    If Not rsNode.EOF And Not rsNode.BOF Then

        Dim iRet
        rsNode.MoveFirst

        Dim pt
        Set pt = New Point

        Do While Not rsNode.EOF
            ' Names
            Dim strName, strShortName
            strName      = CStr( rsNode("Name")  )
            strShortName = Left( rsNode("ShortName"), 8 )

            ' VoltageLevel/NetworkLevel & NetworkGroup
            Dim iLevelID, iGroupID
            iLevelID = GetVoltageLevel( CDbl( rsNode("Un") ) )
            iGroupID = 1

            ' Type of Node
            Dim iType
            iType = GetNodeType( rsNode("NodeType") )

            ' Position & Height
            Dim dSh
            dSh = 1.0
            pt.SetXY CDbl( rsNode("hr") ), CDbl( rsNode("hh") )

            ' Add data of node to internal arrays
            iRet = InsertIntoNodeArray( CStr( rsNode("ID") ), iNodeID, pt, iLevelID )

            if iMode = 1 Then
                Dim iNID
                iNID = AddNode( strName, strShortName, iLevelID, iGroupID, iType, pt.x, …
                If Not pt.IsEmptyPoint Then
                    iRet = AddGraphicNode( iNID, 1, pt, pt )
                End If
```

```
            Else
                iNodeID = iNodeID + 1
            End If

            rsNode.MoveNext
        Loop

        Set pt = Nothing
    End If

    Call CloseRecordset( rsNode )
    Call CloseDatabase()
End Sub
```

Records are selected from the ImportDatabase with following SQL command:

```
Call OpenRecordset( "SELECT Name AS ID, Name, ShortName, NodeType, NetworkLevel, Un, hr, hh "
_
                  & "FROM Node", rsNode )
```

These records are processed in a loop and stored to an internal list using **AddNode**.

```
'-------------------------------------------------------------------------------
' Add Node
'-------------------------------------------------------------------------------
Function AddNode( strName_, strShortName_, iVoltLevelID_, iGroupID_, iType_, dHr_, dHh_, dSh_
)
    Dim iRet

    If bElectro Then
        iRet = InsertIntoArray( arrNode, iCntNode, _
            "insert into " & strTableNode & "( Node_ID, VoltLevel_ID, Group_ID, Name, …
                & iNodeID & "," _
                & iVoltLevelID  & "," _
                & iGroupID  & "," _
                & "'" & strName_ & "'," _
                & "'" & strShortName_ & "'," _
                & iType_ & "," _
                & dHr_ & "," _
                & dHh  & "," _
                & dSh  & "," _
                & "1,1 )" )
    Else
        iRet = InsertIntoArray( arrNode, iCntNode, _
            "insert into " & strTableNode & "( Node_ID, NetworkLevel_ID, Group_ID, Name, …
                & iNodeID & "," _
                & iVoltLevelID  & "," _
                & iGroupID_ & "," _
                & "'" & strName_ & "'," _
                & "'" & strShortName_ & "'," _
                & iType_ & "," _
                & dHr  & "," _
                & dHh  & "," _
                & dSh_ & "," _
                & "1,1 )" )
    End If

    AddNode = iNodeID
    iNodeID = iNodeID + 1
End Function
```

AddNode is actually very simple. At each call, the parameters generate a SQL string that is entered in the **arrNode** array. The SQL commands that are generated look as follows:

```
insert into Node( Node_ID, VoltLevel_ID, Group_ID, Name, ShortName, Flag_Type,
                  hr, hh, sh, Variant_ID, Flag_Variant)
       values ( 1,2,1,'K1','K1',2,7750,21500,1,1,1 )
```

```
insert into GraphicNode( GraphicNode_ID, GraphicLayer_ID, GraphicType_ID,
                         GraphicText_ID1, Node_ID, NodeStartX, NodeStartY, NodeEndX,
NodeEndY,
                         SymType, FrgndColor, BkgndColor, PenStyle, PenWidth, NodeSize, Flag,
                         Flag Variant, Variant ID )
       values ( 1,1,1,1,1,7750,21500,7750,21500,1,0,-1,0,2,4,0,1,1 )

insert into GraphicText( GraphicText_ID, GraphicLayer_ID, Font, FontStyle,
                         FontSize, TextAlign, TextOrient, TextColor, Visible,
                         AdjustAngle, Angle, Pos1, Pos2, Flag, RowTextNo, AngleTermNo,
                         Variant ID, Flag Variant )
       values ( 1,1,'Arial',17,11,3,0,0,1,0,0,0.25,0.25,0,0,0,1,1 )
```

This command show that you do not need to fill in all the table attributes. It is sufficient to fill the key attributes (highlighted above). Basically, these are the primary key, secondary key and the variant code. PSS SINCAL automatically fills in all the other attributes with the default values (if they have not already been filled in).

Read out Line Data from the Database – ReadLines

ReadLines reads out the line data from the sources database and converts this to PSS SINCAL format. As with **ReadNodes**, it generates the appropriate SQL commands when it converts the data. Reading out and processing the network element data is, however, more complicated, since more tables need to be filled as with lines.

```
'-------------------------------------------------------------------------------
' Read Line data from IMPORT db
'-------------------------------------------------------------------------------
Sub ReadLines( iMode ) ' iMode = 1 ... Normal Mode, 0 ... Only init
    Dim rsLine

    Call ReadNodes( 0 )

    WScript.Echo "Reading Lines..."

    Call OpenDatabase(strIMPORTdb)
    Call OpenRecordset( "SELECT Node1 AS Node_ID1, Node2 AS Node_ID2, Name AS ID, Name, " _
     & "'' As ShortName, "0 As Nr, l AS LineLength, r, x, c, Un, " _
     & Ith, 'ON' AS Status, 'ON' AS Switch1, 'ON' AS Switch2 "
     & "FROM Line", rsLine )

    If Not rsLine.EOF and Not rsLine.BOF Then

        Dim iRet
        rsLine.MoveFirst

        Dim iTempNodeID
        iTempNodeID = 0

        Dim Node1, Node2
        Set Node1 = New Node
        Set Node2 = New Node

        Do While Not rsLine.EOF
            Dim bIsVaid
            bIsVaid = True

            Set Node1 = dctNodes.Item( CStr( rsLine("Node_ID1") ) )
            Set Node2 = dctNodes.Item( CStr( rsLine("Node_ID2") ) )

            ' Skip bad objects
            If IsEmpty( Node1 ) Or IsEmpty( Node2 ) Or Node1.iID = Node2.iID Then
                bIsVaid = False
            End If

            If IsNull( rsLine("LineLength") ) Then
                bIsVaid = False
```

```
            End If

            If bIsVaid Then
                ' VoltageLevel/NetworkLevel & NetworkGroup
                Dim iLevelID, iGroupID
                iLevelID = Node1.iLevel
                iGroupID = 1

                ' Check if there are multiple line segments - in this case we must add new
nodes
                Dim strName, strShortName, strFullName, iNr, iCntNr
                strName      = CStr( rsLine("Name") )
                strShortName = Left( rsLine("ShortName"), 8 )
                strFullName  = CStr( rsLine("Node ID1") & "|" & rsLine("Node ID2") & "|" …
                If dctLineSegments.Exists( strFullName ) Then iCntNr = dctLineSegments.Item(
…
                If Not IsNull( rsLine("Nr") ) Then iNr = CLng( rsLine("Nr") ) Else iNr = 0 …

                Dim iInternalID1, iInternalID2
                iInternalID1 = Node1.iID
                iInternalID2 = Node2.iID

                If iCntNr > 1 Then
                    If iNr = 1 Then
                       iTempNodeID = iNodeID
                       iInternalID2 = iTempNodeID
                    ElseIf iNr = iCntNr Then
                       iInternalID1 = iTempNodeID
                       iTempNodeID  = 0
                    Else
                       iInternalID1 = iTempNodeID
                       iTempNodeID  = iNodeID
                       iInternalID2 = iTempNodeID
                    End If
                Else
                    iTempNodeID = 0
                End If

                If iTempNodeID > 0 Then
                    Dim strTempName
                    strTempName = "K" & iNodeID
                    iRet = AddNode( strTempName, strTempName, iLevelID, iGroupID, 1, 0.0, …
                End If

                ' Process standard type mapping
                Dim iStandardType, iFlagStandardType
                iStandardType = 0
                iFlagStandardType = 0

                ' Map the input status of the element
                Dim iState
                iState = GetElementState( rsLine("Status") )

                ' Get Switches
                Dim iTermState1, iTermState2
                iTermState1 = GetSwitchState( CStr( rsLine("Switch1") ) )
                iTermState2 = GetSwitchState( CStr( rsLine("Switch2") ) )

                Dim iEleID, iTermID1, iTermID2
                iEleID   = AddElement( "Line", strName, strShortName, iLevelID, iGroupID, …
                iTermID1 = AddTerminal( iEleID, iInternalID1, 1, 7, iTermState1 )
                iTermID2 = AddTerminal( iEleID, iInternalID2, 2, 7, iTermState2 )

                iRet = InsertIntoArray( arrLine, iCntLine, _
                    "insert into Line ( Element_ID, Typ_ID, Flag_Typ_ID, l, r, x, c, " _
                        & "Un, Ith ) values ( "
                        & iEleID & ","
                        & iStandardType & ","
                        & iFlagStandardType & ","
                        & rsLine("LineLength") & ","  _
                        & rsLine("r") & ","  _
                        & rsLine("x") & ","
                        & rsLine("c") & ","
                        & rsLine("Un") & ","  _
```

```
                        & rsLine("Ith") _
                        & " )" )

                If Not Node1.IsPosEmpty() And Not Node2.IsPosEmpty() Then
                    Dim iGraEleID, iGraTermID1, iGraTermID2
                    iGraEleID   = AddGraphicElement( iEleID, 19, Node1.ptPos, Node2.ptPos )
                    iGraTermID1 = AddGraphicTerminal( iTermID1, iEleID, iGraEleID,
Node1.ptPos )
                    iGraTermID2 = AddGraphicTerminal( iTermID2, iEleID, iGraEleID,
Node2.ptPos )
                End If

          End If

          rsLine.MoveNext
        Loop

        Set Node1 = Nothing
        Set Node2 = Nothing
    End If

    Call CloseRecordset( rsLine )
    Call CloseDatabase()


End Sub
```

The following SQL command reads out the line data from the import database:

```
Call OpenRecordset( "SELECT Node1 AS Node ID1, Node2 AS Node ID2, Name AS ID, Name, "
     & "'' As ShortName, "0 As Nr, l AS LineLength, r, x, c, Un, "
     & Ith, 'ON' AS Status, 'ON' AS Switch1, 'ON' AS Switch2 " _
     & "FROM Line", rsLine )
```

Records selected in this way are processed in a program loop. For each line record, it creates the appropriate SQL insert commands. Basically, the following commands are used:

```
iEleID   = AddElement( "Line", strName, strShortName, iLevelID, iGroupID, 3, iState )
iTermID1 = AddTerminal( iEleID, iInternalID1, 1, 7, iTermState1 )
iTermID2 = AddTerminal( iEleID, iInternalID2, 2, 7, iTermState2 )

iRet = InsertIntoArray( arrLine, iCntLine,
       "insert into Line ( Element ID, Typ ID, Flag Typ ID, l, r, x, c, "
           & "Un, Ith ) values ( " _
           & iEleID & "," _
           & iStandardType & "," _
           & iFlagStandardType & ","
           & rsLine("LineLength") & ","
           & rsLine("r") & ","
           & rsLine("x") & ","  _
           & rsLine("c") & ","  _
           & rsLine("Un") & ","  _
           & rsLine("Ith")
           & " )" )
```

The network diagram for the line is generated with the following commands:

```
iGraEleID   = AddGraphicElement( iEleID, 19, Node1.ptPos, Node2.ptPos )
iGraTermID1 = AddGraphicTerminal( iTermID1, iEleID, iGraEleID, Node1.ptPos )
iGraTermID2 = AddGraphicTerminal( iTermID2, iEleID, iGraEleID, Node2.ptPos )
```

The SQL commands that are generated to create line data look as follows:

```
insert into Element( Element_ID, VoltLevel_ID, Group_ID,
                     Name, ShortName, Type, Flag Input, Flag State,
                     Variant ID, Flag Variant )
       values ( 1,1,1,'L10','','Line',3,1,1,1 )
```

```
insert into Line( Element_ID, Typ_ID, Flag_Typ_ID, l, r, x, c, Un, Ith )
       values ( 1,0,0,2.934134593,0.05,0.21,0,10,0.35 )

insert into Terminal( Terminal ID, Element ID, Node ID,
                      TerminalNo, Flag Terminal, Flag State,
                      Variant ID, Flag Variant )
       values ( 1,1,2,1,7,1,1,1 )

insert into Terminal( Terminal ID, Element ID, Node ID,
                      TerminalNo, Flag Terminal, Flag State,
                      Variant ID, Flag Variant )
       values ( 2,1,4,2,7,1,1,1 )

insert into GraphicElement( GraphicElement ID,GraphicLayer ID,GraphicType ID,
                            GraphicText ID1,Element ID,SymbolDef,
                            FrgndColor,BkgndColor,PenStyle,PenWidth,SymbolSize,
                            SymCenterX,SymCenterY,SymbolType,SymbolNo,Flag,
                            Variant_ID,Flag_Variant)
       values ( 1,1,1,28,1,-1,0,-1,0,1,100,12250,22125,19,0,1,1,1 )

insert into GraphicTerminal( GraphicTerminal ID, GraphicElement ID,
                             GraphicText ID, Terminal ID, PosX, PosY,
                             FrgndColor, PenStyle, PenWidth, SwtType,
                             SwtAlign, SwtNodePos, SwtFactor, SwtFrgndColor,
                             SwtPenStyle, SwtPenWidth, SymbolType, SymbolAlign,
                             SymbolNodePos, SymbolFactor, SymbolFrgndColor,
                             SymbolPenStyle, SymbolPenWidth ,TextAlign, Flag,
                             Variant ID, Flag Variant )
       values ( 1,1,29,1,11250,21500,0,0,1,0,4,20,80,-1,0,1,0,4,40,80,-1,0,1,292,0,1,1 )

insert into GraphicTerminal( GraphicTerminal_ID, GraphicElement_ID,
                             GraphicText ID, Terminal ID, PosX, PosY,
                             FrgndColor, PenStyle, PenWidth, SwtType,
                             SwtAlign, SwtNodePos, SwtFactor, SwtFrgndColor,
                             SwtPenStyle, SwtPenWidth, SymbolType, SymbolAlign,
                             SymbolNodePos, SymbolFactor, SymbolFrgndColor,
                             SymbolPenStyle, SymbolPenWidth ,TextAlign, Flag,
                             Variant ID, Flag Variant )
       values ( 2,1,30,2,13250,22750,0,0,1,0,4,20,80,-1,0,1,0,4,40,80,-1,0,1,292,0,1,1 )

insert into GraphicText( GraphicText_ID, GraphicLayer_ID, Font, FontStyle,
                         FontSize, TextAlign, TextOrient, TextColor, Visible,
                         AdjustAngle, Angle, Pos1, Pos2, Flag, RowTextNo, AngleTermNo,
                         Variant ID, Flag Variant )
       values ( 28,1,'Arial',17,11,3,0,0,1,0,0,0.25,0.25,0,0,0,1,1 )

insert into GraphicText( GraphicText_ID, GraphicLayer_ID, Font, FontStyle,
                         FontSize, TextAlign, TextOrient, TextColor, Visible,
                         AdjustAngle, Angle, Pos1, Pos2, Flag, RowTextNo, AngleTermNo,
                         Variant ID, Flag Variant )
       values ( 29,1,'Arial',17,11,3,0,0,1,0,0,0.25,0.25,0,0,0,1,1 )

insert into GraphicText( GraphicText_ID, GraphicLayer_ID, Font, FontStyle,
                         FontSize, TextAlign, TextOrient, TextColor, Visible,
                         AdjustAngle, Angle, Pos1, Pos2, Flag, RowTextNo, AngleTermNo,
                         Variant ID, Flag Variant )

       values ( 30,1,'Arial',17,11,3,0,0,1,0,0,0.25,0.25,0,0,0,1,1 )
```

Save Read out Data – WriteSINCAL

WriteSINCAL writes data already read out and converted to SQL insert commands to the PSS SINCAL network database.

```
'-------------------------------------------------------------------------------
' Insert data into SINCAL database
'-------------------------------------------------------------------------------
Sub WriteSINCAL()
    WScript.Echo "Writing Data..."

    Call OpenDatabase( strSINCALdb )
```

```
    ' Nodes
    Call InsertRecords( iCntNode, arrNode, "Node: " )

    ' Element & Terminal
    Call InsertRecords( iCntElement,  arrElement,  "Element: " )
    Call InsertRecords( iCntTerminal, arrTerminal, "Terminal: " )

   ' Lines
    Call InsertRecords( iCntLine, arrLine, "Line: " )

   ' Transfomer
    Call InsertRecords( iCntTransformer, arrTransformer, "TwoWindingTransformer: " )

    ' Load & Customer data
    Call InsertRecords( iCntLoad,         arrLoad,         "Load: " )

    ' Graphics
    Call InsertRecords( iCntGraphicNode,     arrGraphicNode,     "GraphicNode: " )
    Call InsertRecords( iCntGraphicText,     arrGraphicText,     "GraphicText: " )
    Call InsertRecords( iCntGraphicTerminal, arrGraphicTerminal, "GraphicTerminal: " )
    Call InsertRecords( iCntGraphicElement,  arrGraphicElement,  "GraphicElement: " )

    Call CloseDatabase()
End Sub
```

3.2 Help Program for Creating PSS SINCAL Database

For your own coupling solutions, you need to fill a blank PSS SINCAL network database with your own data. For this purpose, PSS SINCAL installation has the **SinDBCreate.exe** help program. SinDBCreate lets you create PSS SINCAL network databases as well a standard and protection device databases without the PSS SINCAL user interface.

The program must be started in a command prompt. There is no graphic user interface, i.e. start parameters control the program. Different settings from the PSS SINCAL Registry are also used (e.g. Oracle database configuration), if these have not been entered as parameters.

When you the start the program without parameters, PSS SINCAL displays the following information:

```
C:\> SinDBCreate.exe

Usage:
SinDBCreate /DBSYS:xxx /FILE:xxx /TYPE:xxx [Options]
    Create a new SINCAL-Database.

SinDBCreate /LIST /DBSYS:xxx /ADMIN:User/Password /SRV:xxx
    List all Databases on a server.

SinDBCreate /DELETE /DBSYS:xxx /FILE:xxx /ADMIN:User/Password /SRV:xxx
    Delete a SINCAL-Database on a database server.

    /DBSYS:{ACCESS|ORACLE|SQLSERVER|SQLEXPRESS}
                                Database-System

    /FILE:{Database}
        MS Access:              Path and FileName of the MDB-File
        SQL Server Express:     Path and Filename of the MDF-Datafile
        ORACLE:                 User/Password@Instance
        SQL Server:             Database@Instance

    /ADMIN:User/Password        Administrator-Login for Database-Servers
    /USER:User/Password         Login Information for Database-Servers
    /SRV:Instance               Database Service Name/Server Name

    /TYPE:{E|W|G|H}             Network-Type (E)lectro|(W)ater|(G)as|(H)eating
    [/DB:{NET|STD|PROT}]        Database-Type (Network-Database is default]
    [/DATA]                     Fills STD-DB and Prot-DB with default data
```

```
    [/LANG:{ENG|GER}]           Language for database (default is ENG)
    [/SIN:Filename]             Path and filename of the SIN-file.
```

Create a PSS SINCAL Database

The main function of SinDBCreate is to create PSS SINCAL databases. All the settings needed for this must be indicated as parameters.

```
SinDBCreate /DBSYS:xxx /FILE:xxx /TYPE:xxx [Options]
    Create a new SINCAL-Database.
```

Required Parameters

```
/DBSYS:{ACCESS|ORACLE|SQLSERVER|SQLEXPRESS}
                            Database-System

/FILE:{Database}
    MS Access:              Path and FileName of the MDB-File
    SQL Server Express:     Path and Filename of the MDF-Datafile
    ORACLE:                 User/Password@Instance
    SQL Server:             Database@Instance

/ADMIN:User/Password        Administrator-Login for Database-Servers
/USER:User/Password         Login Information for Database-Servers
/SRV:Instance               Database Service Name/Server Name

/TYPE:{E|W|G|H}             Network-Type (E)lectro|(W)ater|(G)as|(H)eating
```

Optional Parameters

```
[/DB:{NET|STD|PROT}]        Database-Type (Network-Database is default]
[/DATA]                     Fills STD-DB and Prot-DB with default data
[/LANG:{ENG|GER}]           Language for database (default is ENG)
[/SIN:Filename]             Path and filename of the SIN-file.
```

The **DBSYS** parameter determines which database system is used. You can select between **ACCESS** (Microsoft Access), **ORACLE**, **SQLSERVER** (SQL Server) and **SQLEXPRESS** (SQL Server Express).

The **FILE** parameter designates the PSS SINCAL database name. Depending on which database system is used, this parameter must be entered in different ways. In Microsoft Access and SQL Server Express, you need the complete path and file name. In Oracle, you need the user name, password and server name in the format "User/Password/@Server". When you use the SQL Server, you need the database name and server name in the format "DBName@Server".

ADMIN is required for Oracle and SQL Server database systems. This is needed for the main user managing the PSS SINCAL networks. This parameter is defined in the format "User/Password". If this parameter has not been entered, the settings are loaded from the PSS SINCAL registry.

USER is required when the SQL Server is used as a database system. This provides information on who is using the SQL Server and is defined in the format "User/Password".

SRV is used to explicitly enter the database server. If this parameter has not been entered, the server name is loaded from the PSS SINCAL registry.

The **TYPE** parameter indicates the network type. You can select **E** (electricity), **W** (water), **G** (gas) and **H** (district heating/district cooling) networks.

All other parameters are optional and control the generation procedure.

To define the type of database to be created, use the parameter for **DB**. The default is network database **(NET)**. Additional values for these parameters are **STD** for standard type database and **PROT** for protection device database.

The **DATA** parameter fills the standard database or protection-device database with standard types or standard devices.

LANG lets you select the language (English, German) of the network database.

SIN lets you enter the PSS SINCAL network file. This parameter is only for creating network databases.

Example of Creating a Network Database

```
C:\> SinDBCreate.exe /DBSYS:ACCESS /FILE:C:\Temp\dbnet.mdb /TYPE:E
```

The above example creates the Access database "dbnet.mdb" for an electrical network in the "C:\Temp" directory in English.

Example of Creating a Standard Database

```
C:\> SinDBCreate.exe /DBSYS:ORACLE /FILE:OraSTDFL/pwd123@ORA10 /TYPE:W /ADMIN:SINCAL/SINCAL
/DB:STD /DATA
```

In the above example, an Oracle standard database has been created for pipe networks in English. This database has "OraSTDFL" for the Oracle user and the password "pwd123". Additionally the database is filled with standard types.

Example of Creating a Protection Devices Database

```
C:\> SinDBCreate.exe /DBSYS:ACCESS FILE:C:\Temp\stdprot.mdb /TYPE:E /DB:PROT
```

The above example creates an Access protection device database for electrical networks in English. The "stdprot.mdb" database is created in the directory "C:\Temp". An empty database is created.

List PSS SINCAL Databases

In addition to creating PSS SINCAL databases, the SinDBCreate help program can be used to list all the databases at a database server. Switch ON this function with the **LIST** parameter.

```
SinDBCreate /LIST /DBSYS:xxx /ADMIN:User/Password /SRV:xxx
    List all Databases on a server.
```

Required Parameters

```
/DBSYS:{ORACLE|SQLSERVER}
                            Database-System

/ADMIN:User/Password        Administrator-Login for Database-Servers
/SRV:Instance               Database Service Name/Server Name
```

The **DBSYS** parameter determines which database system is used. You can select between **ORACLE** and **SQLEXPRESS** (SQL Server).

ADMIN is required for Oracle and SQL Server database systems. This is needed for the main user managing the PSS SINCAL networks. This parameter is defined in the format "User/Password". If this parameter has not been entered, the settings are loaded from the PSS SINCAL registry.

SRV is used to explicitly enter the database server. If this parameter has not been entered, the server name is loaded from the PSS SINCAL registry.

Example

```
C:\> SinDBCreate /LIST /DBSYS:ORACLE /ADMIN:SINCAL/SINCAL /SRV:ORA10
```

The above example lists all available PSS SINCAL databases. First of all a connection to the ORACLE instance with the name "ORA10" and the user "SINCAL" is established. If the connection is successful the available databases are displayed line by line.

Delete a PSS SINCAL Database

In addition to creating PSS SINCAL databases, the SinDBCreate help program can be used to delete a PSS SINCAL database at a database server. Switch this function ON with the **DELETE** parameter.

```
SinDBCreate /DELETE /DBSYS:xxx /FILE:xxx /ADMIN:User/Password /SRV:xxx
    Delete a SINCAL-Database on a database server.
```

Required Parameters

```
/DBSYS:{ORACLE|SQLSERVER}
                            Database-System

/FILE:{Database}
    ORACLE:                 User
    SQL Server:             Database

/ADMIN:User/Password        Administrator-Login for Database-Servers
/SRV:Instance               Database Service Name/Server Name
```

The **DBSYS** parameter determines which database system is used. You can select between **ACCESS** (Microsoft Access), **ORACLE**, **SQLSERVER** (SQL Server) and **SQLEXPRESS** (SQL Server Express).

The **FILE** parameter designates the PSS SINCAL database name. Depending on which database system is used, this parameter must be entered in different ways. When using Oracle, enter the user name. When using the SQL Server, enter the database name.

ADMIN is required for Oracle and SQL Server database systems. This is needed for the main user managing the PSS SINCAL networks. This parameter is defined in the format "User/Password". If this parameter has not been entered, the settings are loaded from the PSS SINCAL registry.

SRV is used to explicitly enter the database server. If this parameter has not been entered, the server name is loaded from the PSS SINCAL registry.

Example

```
C:\> SinDBCreate /DELETE /DBSYS:ORACLE /FILE:SINCAL_TEST /ADMIN:SINCAL/SINCAL /SRV:ORA10
```

The above example deletes the PSS SINCAL "SINCAL_TEST" database. First of all a connection to the ORACLE instance with the name "ORA10" and the user "SINCAL" is established. Deleting the PSS SINCAL database cannot be undone.

4 Automation of the Calculation Methods

The PSS SINCAL architecture is based on a system of different components that use COM functions to communicate (see Reference – PSS SINCAL Architecture). This chapter explains how the calculation methods can be integrated into own applications with the help of COM functions.

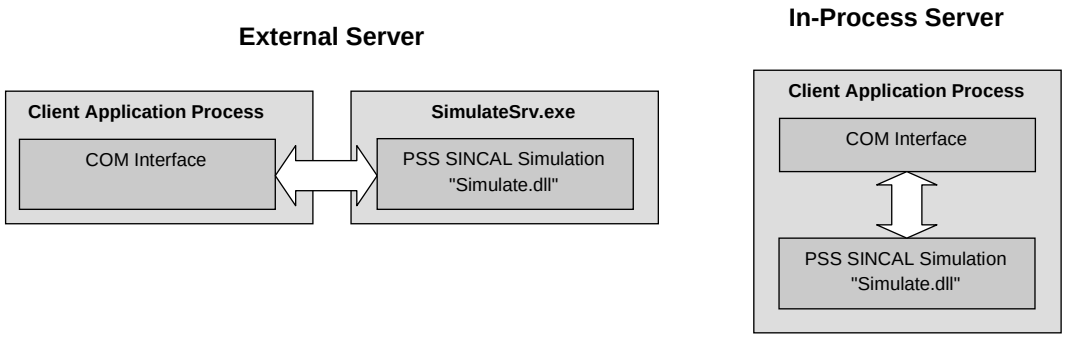
Excerpts of codes and examples are explained with the help of the **Windows Scripting Host (WSH)**, since this has the simplest syntax and is normally available directly in the current operating systems. Any programming language that supports COM functions (e.g. VisualBasic, VBA, C++, etc.) can be used for the automation functions.

The calculation methods' open design can help you solve a variety of different problems. Basically, however, you need to differentiate between **integration into external applications** and the **use in own solutions**.

Integrating the Calculation Methods into External Applications

This type of automation is primarily used for integrated solutions in GIS, NIS or SCADA systems. You start the PSS SINCAL calculations directly from the respective source system. All the data maintenance and processing and the visualization of the results is done directly in the source system.

This automation solution integrates the calculation methods directly into the source system. They are connected by COM interface, where the calculation methods can be used either as an **External Server** (with separate processes) or as an **In-Process Server** (within the same process).



For high performance solutions, you can administer all the network data and, of course, the results in "virtual tables" directly in the PSS SINCAL calculations. The structure of these virtual tables is exactly like the PSS SINCAL data model. COM interfaces write network data, get results and control the calculation methods.

For a C++ sample program that shows how calculation methods are integrated into external applications and how virtual tables are used, see the PSS SINCAL installation structure ("Batch\SimAuto.zip").

Integrating the Calculation Methods in own Solutions

The idea is to use PSS SINCAL calculation methods (load flow, short circuit, etc.) as a basis for individual solutions and analyses. Normally, you use an existing PSS SINCAL network and then analyze this with own solutions.

A simple example: In a network, you need to examine the effect of a load increase. You increase a load value step by step and check the voltage level at the node at the same time.

For this problem, the small sample program "VoltageDrop.vbs" in the PSS SINCAL installation structure is available.

```
'-------------------------------------------------------------------------------
' File:        VoltageDropBatch.vbs
' Description: Small sample for simulation automation.
'              A load at a node is constatly increased  until a specified
'              voltage drop occurs.
' Author:      SS, GM
' Modified:    14.03.2008
'-------------------------------------------------------------------------------
Option Explicit

const siSimulationOK = 1101

Dim strDatabase ' Database of sincal network
strDatabase = "D:\Network\ Test\Example Ele.mdb"

Dim strProtDatabase ' Database with protection devices
strProtDatabase = "D:\Server-Setup\Database\ProtectionDB.mdb"

Dim strLoad ' Name of Load to be changed
strLoad = "LO8"

Dim strLF ' Load flow procedure
strLF = "LF_NR"

' Set locale to US -> necessary because '.' is required for SQL commands!
SetLocale( "en-gb" )

'-------------------------------------------------------------------------------
' Start of the script
'-------------------------------------------------------------------------------
If Not UCase( Right(WScript.Fullname,11) ) = "CSCRIPT.EXE" Then
    Call Usage()
    WScript.Quit
End If

' Create an simulation object as "in process server"
Dim SimulateObj
Set SimulateObj = WScript.CreateObject( "Sincal.Simulation" )

If SimulateObj is Nothing Then
    WScript.Echo "Error: CreateObject Sincal.Simulation failed!"
    WScript.Quit
End If

' Setting databases and language
SimulateObj.DataSourceEx "DEFAULT", "JET", strDatabase, "Admin", ""
SimulateObj.DataSourceEx "PROT", "JET", strProtDatabase, "Admin", ""
SimulateObj.Language "US"

' Enable simulation batch mode: load from phys. database, store to virtual database
SimulateObj.BatchMode 1

' Load from database and generating calculation objects
SimulateObj.LoadDB CStr( strLF )

' Getting calculation object load for modifying
Dim LoadObj
```

```
Set LoadObj = SimulateObj.GetObj( "LOAD", CStr( strLoad ) )
If LoadObj is Nothing Then
    WScript.Echo "Error: Load " & strLoad & " not found!"
    WScript.Quit
End If

' Getting calculation object node of load
Dim NodeID
NodeID = LoadObj.Item( "TOPO.NODE1.DBID" )
Dim LoadNode
Set LoadNode = SimulateObj.GetObj( "NODE", NodeID )

' Getting virtual database object
Dim SimulateNetworkDataSource
Set SimulateNetworkDataSource = SimulateObj.DB EL
If SimulateNetworkDataSource Is Nothing Then
    WScript.Echo "Error: getting virtual database object failed!"
    WScript.Quit
End If

'-------------------------------------------------------------------------------
' Perform special voltage drop analysis
'-------------------------------------------------------------------------------
Dim iLoop, iLoopErr
iLoop = 0
iLoopErr = 0

WScript.Echo vbCrLf & "Start load flow calculation (" & strLF & ")"

Do While iLoop < 1000
    WScript.Echo vbCrLf & "-------- " & CStr( iLoop ) & " --------"

    ' We modify the load by adding 0.1 MW in each loop
    Call ModifyLoad( LoadObj, 0.1 )

    ' Start loadflow simulation
    SimulateObj.Start strLF
    If SimulateObj.StatusID <> siSimulationOK Then
        WScript.Echo "Load flow failed!"
        Exit Do
    End If

    ' Getting load flow result for node
    If LoadNode Is Nothing Then
    Else
        Dim LFNodeResultLoad
        Set LFNodeResultLoad = LoadNode.Result( "LFNODERESULT", 0 )
        If LFNodeResultLoad Is Nothing Then
        Else
            Dim u_un
            u_un = LFNodeResultLoad.Item( "U Un" )
            WScript.Echo "Node voltage at modified load U/Un = " & FormatNumber( u_un ) & "%"
            Set LFNodeResultLoad = Nothing
        End If
    End If

    ' Display some golbal result information
    Call OutputLFAccurResult( SimulateNetworkDataSource )

    iLoop = iLoop + 1
Loop

' Write calculation messages
Call WriteMessages( SimulateObj )

' Release used objects
Set SimulateNetworkDataSource = Nothing
Set LoadObj = Nothing
Set LoadNode = Nothing
Set SimulateObj = Nothing
Set SimulateObj = Nothing

'-------------------------------------------------------------------------------
' Modify load
```

```
'-------------------------------------------------------------------------------
Sub ModifyLoad( ByRef LoadObj_, ValAdd )

    ' Modify load by increasing P and Q
    Dim P
    P = LoadObj .Item( "P" ) + ValAdd
    LoadObj_.Item( "P" ) = P

    Dim Q
    Q = LoadObj .Item( "Q" ) + ValAdd
    LoadObj .Item( "Q" ) = Q

    WScript.Echo "Set load " & strLoad & " to P = " & P & "MW, Q = " & Q  & "Mvar"

End Sub

'-------------------------------------------------------------------------------
' Output some data of LFAccurResult to console
'-------------------------------------------------------------------------------
Sub OutputLFAccurResult( ByRef SimulateNetworkDataSource )

    ' Get datbase object LFAccurResult from virtual database
    Dim LFAccurResult
    Set LFAccurResult = SimulateNetworkDataSource.GetRowObj( CStr( "LFAccurResult" ) )
    If LFAccurResult Is Nothing Then
        WScript.Echo "Error: cant get objects in LFAccurResult!"
        WScript.Quit
    End If

    ' Open table LFAccurResult
    Dim hr
    hr = LFAccurResult.Open
    If hr <> 0 Then
        WScript.Echo "Error: cant open LFAccurResult!"
        WScript.Quit
    End If

    ' Move cursor to first row
    Dim bRead next data
    bRead_next_data = LFAccurResult.MoveFirst

    If bRead_next_data = 0 Then
        ' Get attribut Iteration Number
        Dim IterCnt
        IterCnt = LFAccurResult.Item( "IT" )

        ' Get attribute Power Node Balance
        Dim PNB
        PNB = LFAccurResult.Item( "PNB" )

        ' Get attribute Power Node Balance
        Dim PNBre
        PNBre = LFAccurResult.Item( "PNBre" )

        'Get attribute Voltage Mesh Balance
        Dim VLB
        VLB = LFAccurResult.Item( "VLB" )

        'Get attribut Voltage Mesh Balance
        Dim VLBre
        VLBre = LFAccurResult.Item( "VLBre" )

        'Output to console
        WScript.Echo "IT = " & IterCnt & ", Power Accuracy PNBre = " & FormatNumber( PNBre /
1000.0 ) & "kW"
    End If

    ' Release datbase object LFAccurResult
    Set LFAccurResult = nothing

End Sub

'-------------------------------------------------------------------------------
' Write simulation messages
```

```
'-------------------------------------------------------------------------------
Sub WriteMessages( ByRef SimulateObj )

    WScript.Echo vbCrLf & "Simulation Messages:" & vbCrLf


    Dim objMessages
    Set objMessages = SimulateObj.Messages

    Dim strType
    Dim intMsgIdx
    For intMsgIdx = 1 To objMessages.Count
        Dim Msg
        Set Msg = objMessages.Item( intMsgIdx )

        Select Case Msg.Type
            case 1 ' STATUS
            case 2 ' INFO
            case 3 ' WARNING
                WScript.Echo Msg.Text
            case 4 ' ERROR
                WScript.Echo Msg.Text
        End Select

        Set Msg = Nothing
    Next
    Set objMessages = Nothing
End Sub

'-------------------------------------------------------------------------------
' Show usage
'-------------------------------------------------------------------------------
Sub Usage()
    Dim strUsage
    strUsage = "Usage: cscript.exe VoltageDropBatch.vbs" _
             & vbCrLf & vbCrLf _
             & "A load at a node is constatly increased until a specified " 
             & "voltage drop occurs."
             & vbCrLf
    WScript.Echo strUsage
End Sub
```

Start the sample program at the prompt as follows:

```
> cscript.exe VoltageDrop.vbs
```

After the program starts, PSS SINCAL normally displays an error message. The reason for this is simple. The sample program statically stores the PSS SINCAL network database to be used in the calculations. These global preliminary settings need to be adapted for individual experiments.

```
Dim strDatabase ' Database of sincal network
strDatabase = "D:\Network\_Test\Example Ele.mdb"

Dim strProtDatabase ' Database with protection devices
strProtDatabase = "D:\Server-Setup\Database\ProtectionDB.mdb"

Dim strLoad ' Name of Load to be changed
strLoad = "LO8"
```

First, modify the contents of the **strDatabase** variable. Enter the network to be calculated.

Then change the **strProtDatabase** variable. This defines the database for global protection devices and can be found in the PSS SINCAL Installation "database" directory.

The **strLoad** variable specifies the load to be increased. In our example, this is the load with the name "L08".

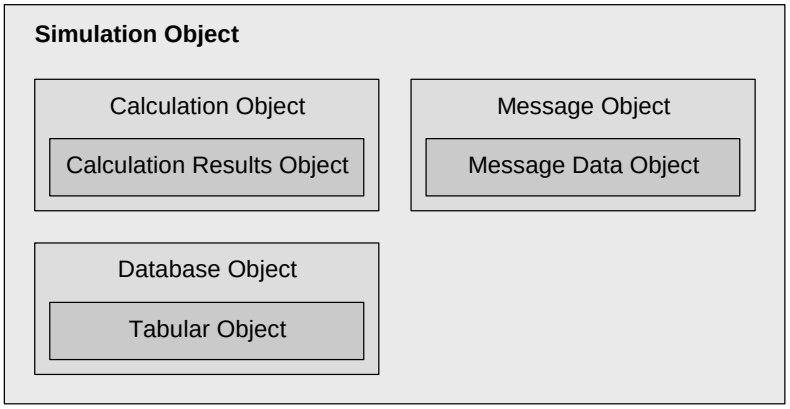
When you restart the sample program after you have modified the database paths, PSS SINCAL displays the following text.

```
>cscript.exe VoltageDrop.vbs

-------- 0 --------
Set load LO8 to P = 0.5MW, Q = 0.4Mvar
Node voltage at modified load U/Un = 92.37%
IT = 9, Power Accuracy PNBre = 0.00kW

-------- 1 --------
Set load LO8 to P = 0.6MW, Q = 0.5Mvar
Node voltage at modified load U/Un = 91.75%
IT = 9, Power Accuracy PNBre = 0.00kW

-------- 2 --------
Set load LO8 to P = 0.7MW, Q = 0.6Mvar
Node voltage at modified load U/Un = 91.13%
IT = 10, Power Accuracy PNBre = 0.00kW

...

-------- 33 --------
Set load LO8 to P = 3.8MW, Q = 3.7Mvar
Node voltage at modified load U/Un = 55.96%
IT = 10, Power Accuracy PNBre = 0.00kW

-------- 34 --------
Set load LO8 to P = 3.9MW, Q = 3.8Mvar
Load flow failed!

Simulation Messages:

W 2714: Element data not physically meaningful
E 3101: Load flow: no convergence break after 200 iterations
E 1070: Please have a look at System Manual - Technical Reference - Messages from
Calculations - Errors for further error information
```

Now the program executes without errors. 34 load flow calculations are performed. Before every calculation, the load value of "L08" increases by 0.1 MW. The 33rd time around, a convergence with a node voltage of 55.96 was still possible. The 34th time, no convergence is possible. This means the maximum load value permitted is P = 3.8 MW and Q = 3.7 Mvar.

4.1 Available Automation Functions

Automation objects in the calculation methods are structured hierarchically. Objects are instantiated from the higher-level object to the lower-level objects. The objects themselves have different methods and functions.



Overview of the Available Automation Functions

Simulation Object:

- BatchMode: Activate virtual database
- DataSourceEx: Set the databases
- Database: Set the databases
- SQLUser: Set the SQL user
- DataFile: Set the data file
- MacroPath: Paths for models
- Language: Set the language
- Currency: Set the currency
- SetInputState: Set the input state
- LoadDB: Load the input data from database
- SaveDB: Save the results to database
- AddObjID: Add Objects
- Parameter: Set and query global parameters
- DoCommand: Execute commands
- Start: Start calculation
- StatusID: Status code of calculation procedure
- GetObj: Access calculation objects
- GetObjById, GetObjByGUID: Access calculation objects using the ID
- DB_EL, DB_FLOW: Access database objects
- Messages: Access message objects

Calculation Object:

- Count: Number of possible attributes
- Name: Determine attribute names
- Item: Access attributes
- Result: Access calculation results object

Calculation Results Object:

- Count: Number of possible attributes
- Name: Determine attribute names
- Item: Access attributes

Database Object:

- GetRowObj: Determine instance for a tabular object

Tabular Object:

- Open: Open the tabular object
- Close: Close a tabular object
- CountRows: Determine the number of records
- MoveFirst, MoveLast, MoveNext, MovePrev: Position in the data
- Count: Number of possible attributes
- Name: Determine attribute names
- Item: Access attributes

Message Object:

- Count: Number of possible messages
- Item: Access message data object

Message Data Object:

- Text: Message text
- Type: Message type
- CountObjectIds: Number of network elements
- ObjectIdAt, ObjectTypeAt: Network element data

Attributes of Calculation Objects:

- Attributes for electrical networks
- Attributes for pipe networks

4.1.1 Simulation Object

This object creates the basis for all further automation functions. It is a duplicate of the PSS SINCAL calculation module and, as such, is the main object of any automation. Simulation objects can be instantiated as either **Local Servers** or **In-Process Servers**.

Local Server

Local servers are executable programs that implement COM components. When a COM component is instantiated, this program starts as its own background process.

For communication between the processes, PSS SINCAL uses a special RPC log (Remote Procedure Call) to slow down the speed in the data exchange. The advantage is that completely separate processes and storage models are used. This means that even serious program errors do not influence any other processes.

The following starts the calculations as a new process.

```
Set SincalSimSrv = WScript.CreateObject( "Sincal.SimulationSrv" )
If SincalSimSrv Is Nothing Then
    WScript.Echo "Error: CreateObject Sincal. SincalSimSrv failed!"
    WScript.Quit
End If
```

The simulation object is accessed with the COM interface.

```
Set SincalSim = SincalSimSrv.GetSimulation
If SincalSim Is Nothing Then
    WScript.Echo "Error: GetSimulation failed!"
    WScript.Quit
End If
```

In-Process Server

In the case of an in-process server, PSS SINCAL provides the interfaces in a DLL. If you instantiate a COM component of an in-process server, PSS SINCAL loads the appropriate server into the current process. In-process servers are particularly fast, since the functions of the interfaces are accessed within the limits of the process.

The following instantiates the calculations in the "current" process as an additional COM component.

```
Set SincalSim = WScript.CreateObject( "Sincal.Simulation" )
If SincalSim Is Nothing Then
    WScript.Echo "Error: CreateObject Sincal.Simulation failed!"
    WScript.Quit
End If
```

BatchMode – Activate Virtual Database

Changes the database mode of the calculations.

```
SimulateObj.BatchMode iMode
```

Parameters

iMode (Integer)
Database mode for the calculations. The mode is a numerical value from 0 to 2.

Code	Description
0	Load from a real database, save in a real database
1	Load from a real database, save in a virtual database
2	Load from a virtual database, save in a virtual database
4	Load from a real in a virtual database, save in a virtual database

Comments

In normal simulation, the output data are stored directly in the database. The **Batchmode** function, however, sends the output data to a virtual database. This greatly increases the speed, since you do not lose time making entries in the database. The results are kept only in the virtual database of the main memory for the calculations.

Example

```
' Enable virtual database.
SimulateObj.BatchMode 1
```

DataSourceEx – Set the Databases

Defines the databases used in the calculations.

```
SimulateObj.DataSourceEx strDBType, strDBSystem, strDatabase, strUser, strPassword
```

Parameters

strDBType (String)
Predefined sign for the database type.

Database type	Description
"DEFAULT"	Network database
"PROT"	Global protection device database
"PROT_USR"	Local protection device database

strDBSystem (String)
Predefined sign for the database system.

Database system	Description
"JET"	Microsoft Access
"ORACLE"	Oracle

strDatabase (String)
Complete path and file name of the database.

strUser (String)
User name for the database.

strPassword (String)
Password for the database.

Comments

This function is obsolete and no longer recommended for use. It is only included for reasons of compatibility and should be replaced by the **Database – Set the Databases** function.

Example

```
' Set database filename and path.
SimulateObj.DataSourceEx "DEFAULT", "JET", strDatabase, "Admin", ""
SimulateObj.DataSourceEx "PROT", "JET", strProtDatabase, "Admin", ""
```

Database – Set the Databases

Defines the databases used in the calculations.

```
SimulateObj.Database strConnection
```

Parameters

strConnection (String)
Database connection.

Code	Description
"TYP"	Database type
"MODE"	Database system
"INSTANCE"	Database server
"NAME"	Database name
"USR"	User name
"PWD"	Password
"SYSUSR"	PSS SINCAL administration user
"SYSPWD"	Password of the PSS SINCAL administration user
"FILE"	File name of database
"SINFILE"	File name of PSS SINCAL file

The "TYP" code can have the following values.

Database type	Description
"NET"	Network database
"PROT"	Global protection device database
"PROT_USR"	Local protection device database
"STD"	Global standard database
"STD_USR"	Local standard database

The "MODE" code can have the following values.

Database system	Description
"JET"	Microsoft Access
"ORACLE"	Oracle
"SQLSERVER"	SQL Server
"SQLEXPRESS"	SQL Server Express

Comments

Database connections can either be entered as **pairs** with field=value and ";" or as short forms in the complete sequence.

Pairs:

```
TYP=NET;MODE=JET;...
```

Short form:

```
TYP;MODE;FILE;INSTANCE;NAME;USR;PWD;SINFILE;SYSUSR;SYSPWD;
```

Example

Depending on the database system, different connection codes are used.

```
' Set database connection string depending on database system.
' ACCESS:
SimulateObj.Database "TYP=NET;MODE=JET;FILE=C:\Temp\Example
Ele files\database.mdb;USR=Admin;PWD=;SINFILE=C:\Temp\Example Ele.sin;"

SimulateObj.Database "NET;JET;C:\Temp\Example Ele_files\database.mdb;;;Admin;;C:\Temp\Example
Ele.sin;;;"

' ORACLE:
SimulateObj.Database
"TYP=NET;MODE=ORACLE;USR=ORA_ELE1;PWD=ORA_ELE1;INSTANCE=ORA11;SINFILE=C:\Temp\Example
Ele.sin;SYSUSR=sincal;SYSPWD=sincal;"

SimulateObj.Database "NET;ORACLE;;ORA11;;ORA ELE1;ORA ELE1;C:\Temp\Example
Ele.sin;sincal;sincal;"

' SQLEXPRESS:
SimulateObj.Database "TYP=NET;MODE=SQLEXPRESS;FILE=C:\Temp\Example
Ele_files\database.mdf;NAME=Example Ele;SINFILE=C:\Temp\Example Ele.sin;"

SimulateObj.Database "NET;SQLEXPRESS;C:\Temp\Example Ele files\database.mdf;;Example Ele;;;
C:\Temp\Example Ele.sin;;;"
```

```
' SQLSERVER:
SimulateObj.Database
"TYP=NET;MODE=SQLSERVER;NAME=SQLSRV ELE1;INSTANCE=SQLSRV;USR=username;PWD=password;SINFILE=C:
\Temp\Example Ele.sin;SYSUSR=sincal;SYSPWD=sincal;"

SimulateObj.Database "NET;SQLSERVER;;SQLSRV;SQLSRV ELE1;username;password;C:\Temp\Example
Ele.sin;sincal;sincal;"
```

SQLUser – Set the SQL User

Defines the user name and the password for the SQL Server.

```
SimulateObj.SQLUser strUser, strPassword
```

Parameters

strUser (String)
SQL user name.

strPassword (String)
SQL password.

Example

```
' Set the SQL user.
SimulateObj.SQLUser "User", "Password"
```

DataFile – Set the Data File

Defines the data file needed by import or export.

```
SimulateObj.DataFile strDataFile
```

Parameters

strDataFile (String)
Complete path and file name of the data file.

Example

```
' Set the datafile for imports or exports.
SimulateObj.DataFile "C:\Test\CIM\CIMExample.xml"
```

MacroPath – Paths for Models

Defines the local and global path, from which models are used.

```
SimulateObj.MacoPath strGlobalPath, strLocalPath
```

Parameters

strGlobalPath (String)
Complete path of the directory, where global models are stored.

strLocalPath (String)
Complete path of the directory, where local models are stored.

Example

```
' Set global and local path for models.
SimulateObj.MacroPath "C:\GlobalMacros", "C:\LocalMacros"
```

Language – Set the Language

Determines the language for the calculations and calculation messages.

```
SimulateObj.Language strLanguage
```

Parameters

strLanguage (String)
Predefined code for the language to be set.

Code	Description
"DE"	Output language: German
"US"	Output language: English

Example

```
' Select language for messages.
SimulateObj.Language "DE"
```

Currency – Set the Currency

Determines the currency sign for the calculations and the calculation results.

```
SimulateObj.Currency strCurrency
```

Parameters

strCurrency (String)
Currency sign to be set.

Example

```
' Set currency.
SimulateObj.Currency "EUR"
```

```
SimulateObj.Currency "€"
```

SetInputState – Set the Input State

Sets the input status for the data considered in the calculations.

```
SimulateObj.SetInputState lInputMask
```

Parameters

lInputMask (Long Integer)
Bitwise form with predefined codes for input states.

Code	Description
Electrical networks	
0x00000001	Load Flow
0x00000002	Short Circuit
0x00000004	Harmonics
0x00000008	Motor Start-Up
0x00000010	Low-Voltage Dimensioning
0x00000020	Multiple Faults
0x00000040	Protection
0x00000080	Distance Protection
0x00000100	Optimization
0x00000200	Dynamics
0x00000400	Unbalanced Load Flow
0x00000800	Reliability
0x00001000	Economic Efficiency
0x00002000	Load Assignment
0x00004000	Arc Flash
Pipe networks	
0x00080000	Steady-State Calculations
0x00100000	Dynamic Calculations
0x00200000	Geo-Stationary Calculations

Example

```
const CalcMethod_LF = &H00000001
const CalcMethod_SC = &H00000002

' Set input data mask.
SimulateObj.SetInputState CalcMethod_LF Or CalcMethod_SC
```

LoadDB – Load the Input Data from Database

Loads the database to the main memory and creates the network model for the calculations.

```
SimulateObj.LoadDB strMethod
```

Parameters

strMethod(String)
Predefined sign for the calculation method. For a complete list of permissible values, see the function **Start – Start Calculation**.

Comments

When you select a specific calculation method, PSS SINCAL loads only the data required by this calculation method from the database.

Example

```
' Load input data for load flow calculations from database.
SimulateObj.LoadDB "LF_NR"
```

SaveDB – Save the Results to Database

Stores the virtual results in the physical database so these are available for additional evaluations in the database after the automation solution.

```
SimulateObj.SaveDB strMethod
```

Parameters

strMethod (String)
Predefined sign for the calculation method. For a complete list of permissible values, see the function **Start – Start Calculation**.

Example

```
' Save results to database.
SimulateObj.SaveDB "LF_NR"
```

AddObjID – Add Objects

Defines objects for the calculation method for further special editing. The kind of editing depends on the respective calculation method as well as the control setting.

```
SimulateObj.AddObjID( lRowType, lDBID, eMode )
```

Parameters

lRowType (Long Integer)
Database type of the object.

lDBID (Long Integer)
Database ID of the object.

eMode (Enum)
Predefined code for using objects in the calculations.

Enumeration	Code	Description
ADDOBJ_LF_NONE	0	
ADDOBJ_LF_MALF	1	Contingency Analysis
ADDOBJ_OBJ_SC	2	Short Circuit at Object
ADDOBJ_OBJ_EXP	3	Netomac Export – Determining Machines
ADDOBJ_OBJ_MOT_SIMPLE	4	Simple Motor Start-Up
ADDOBJ_LF_ALLOC	5	Load Allocation
ADDOBJ_LF_RESUP	6	Restoration of Supply
ADDOBJ_FLOW_H2O_MALF	7	Contingency Analysis Water
ADDOBJ_FLOW_GAS_MALF	8	Contingency Analysis Gas
ADDOBJ_FLOW_HEAT_MALF	9	Contingency Analysis Heating/Cooling
ADDOBJ_OPT_CAP	10	Capacitor Placement
ADDOBJ_GEN_PV	11	PV Curves
ADDOBJ_FLOW_H2O_LEAK	12	Fire Water
ADDOBJ_LF_MALF_RECON	13	Contingency Analysis – Reconnection
ADDOBJ_OPT_NET	14	Optimal Network Structure
ADDOBJ_ECO	15	Economic Efficiency
ADDOBJ_NETRED_INCLUDE	16	Not currently used
ADDOBJ_NETRED_EXCLUDE	17	Elements not to be reduced
ADDOBJ_NETRED_BOUNDARY	18	Boundary lines for network reduction

Example

```
' Set object for further usage.
const ADDOBJ_OBJ_SC = 2
Simulation.AddObjID( 4, 1, ADDOBJ_OBJ_SC );
```

Parameter – Set and Query Global Parameters

Sets a global parameter for the calculation method.

```
SimulateObj.Parameter( strParameter ) = Value
Value = SimulateObj.Parameter( strParameter )
```

Properties

Parameter (Variant)
Value of the parameter.

Parameters

strParameter (String)
Predefined name of the parameter. Different calculation methods have different parameters.

Global Parameters

Parameter	Data type	Description
"Sim.Identification"	String	Name identifying the objects "Name" = identification by a name "ShortName" = identification by a short name Determines whether the name ("Name") or short name ("ShortName") is used to identify an object.
"GRAPHIC_VIEWID"	Integer	The graphic view using for the calculations (GraphicAreaTile_ID)

Contingency Analysis

Parameter	Data type	Description
"CA_MODE"	String	Specifies the calculation mode "NORMAL" = normal calculation "REDUCED" = reduced calculation "PRE_ANALYSE" = preanalysis
"CA_PRE_ANALYSE_MODE"	String	Specifies the calculation method for the pre-analysis "VOLT" = voltage change "ISOL_POWER" = not delivered power "ISOL_ELEMENTS" = not supplied elements "ISOL_CONSUMERS" = not supplied loads
"CA_PRE_ANALYSE_COUNT"	Integer	Number of malfunctions to be calculated

Restoration of Supply

Parameter	Data type	Description
"LF_RESUP_MODE"	Integer	Specifies the mode 0 = standard 1 = feeder based
"LF_RESUP_RESUPPLYCNT_ACT"	Integer	Activation of the maximum number of restorations of supply 0 = not activated 1 = activated
"LF_RESUP_RESUPPLYCNT"	Integer	Maximum number of resupplies
"LF_RESUP_SWITCHCNT_ACT"	Integer	Activation of the maximum number of switching actions 0 = not activated 1 = activated
"LF_RESUP_SWITCHCNT"	Integer	Maximum number of switching actions
"LF_RESUP_LOADSHEDDING_ACT"	Integer	Activation of load shedding 0 = not activated 1 = activated
"LF_RESUP_VIOLATION_ACT"	Integer	Activation of limit violations 0 = not activated 1 = activated
"LF_RESUP_SWITCHCNT_FACTOR"	Double	Weighting for switching actions
"LF_RESUP_LOADSHEDDING_FACTOR"	Double	Weighting for load shedding
"LF_RESUP_VIOLATION_FACTOR"	Double	Weighting for violations

VoltVar

Parameter	Data type	Description
"OPT_VOLTVAR_CAP_SN"	Double	Rated apparent power for the capacitor
"OPT_VOLTVAR_TRAFO"	Integer	Activation of the transformer 0 = not activated

		1 = activated
"OPT_VOLTVAR_TRAFO_SN"	Double	Rated apparent power for the transformer
"OPT_VOLTVAR_TRAFO_UK"	Double	Reference short circuit voltage for the transformer
"OPT_VOLTVAR_LIMIT_LOWER"	Double	Voltage lower limit in %
"OPT_VOLTVAR_LIMIT_UPPER"	Double	Voltage upper limit in %
"OPT_VOLTVAR_MINMAX_MODE"	Integer	Specifies the calculation mode 0 = factor 1 = operating point
"OPT_VOLTVAR_MIN_FACTOR"	Double	Factor for minimum
"OPT_VOLTVAR_MAX_FACTOR"	Double	Factor for maximum
"OPT_VOLTVAR_MIN_OPID"	Integer	Operating point for minimum
"OPT_VOLTVAR_MAX_OPID"	Integer	Operating point for maximum

Static Network Reduction

Parameter	Data type	Description
"STATNETRED_USESOURCEDB"	Integer	Defines whether the original database is to be changed or the reduced network written to a second database. 0 = Fill second database with reduced network 1 = Carry out change of the original database
"STATNETRED_SINFILE"	String	Complete file name of the SIN file of the second database. e.g.: "D:\Network\Red-RS.sin"
"STATNETRED_DATABASE"	String	Database definition for the second database. e.g.: "TYP=NET;MODE=JET;FILE=D:\Network\Red-RS_files\database.mdb;USR=Admin;SINFILE=D:\Network\Red-RS.sin;"
"STATNETRED_CREATEGRAPHIC"	Integer	Activates the graphic generation of the boundary nodes when using two separate databases. 0 = Do not generate a graphic 1 = Generate graphic
"STATNETRED_EXTWARD"	Integer	Activates the determination of boundary injections with the Extended Ward procedure. 0 = Do not generate Extended Wards 1 = Generate Extended Wards
"STATNETRED_SC1"	Integer	Activates the determination of the zero-phase sequence data in the reduced network for asymmetrical short circuit calculations. 0 = Do not determine any zero-phase sequence data 1 = Determine zero-phase sequence data
"STATNETRED_SC3"	Integer	Activates the determination of short circuit data in the reduced network. 0 = Do not determine any short circuit data 1 = Determine short circuit data

Dynamic Network Reduction

Parameter	Data type	Description
"DYNNETRED_USESOURCEDB"	Integer	Defines whether the original database is to be changed or the reduced network written to a second database. 0 = Fill second database with reduced network 1 = Carry out change of the original database
"DYNNETRED_SINFILE"	String	Complete file name of the SIN file of the second database. e.g.: "D:\Network\Red-RS.sin"
"DYNNETRED_DATABASE"	String	Database definition for the second database. e.g.: "TYP=NET;MODE=JET;FILE=D:\Network\Red-RS_files\database.mdb;USR=Admin;SINFILE=D:\Network\Red-RS.sin;"
"DYNNETRED_CREATEGRAPHIC"	Integer	Activates the graphic generation of the boundary nodes when

		using two separate databases: 0 = Do not generate a graphic 1 = Generate graphic
"DYNNETRED_TIMESTART"	Double	Start time in seconds for the correlation functions
"DYNNETRED_TIMEEND"	Double	End time in seconds for the correlation functions
"DYNNETRED_LOWERLIMIT"	Double	Lower limit value for the correlation factor
"DYNNETRED_MACHINES"	Integer	Number of coherent machines to be generated in the reduced network. If "0" is entered, the number is determined automatically.
"DYNNETRED_FUNCTION"	Integer	Determines which signal is used for the correlation functions. 1 = slip 2 = load angle 3 = active power 4 = reactive power 5 = voltage It is a good idea to always select "Slip" as this provides the best results.
"DYNNETRED_REFNODE"	Integer	NodeID of the reference node in the subnetwork to be reduced
"DYNNETRED_REFVOLTAGE"	Double	Ref. voltage for net equivalent in kV
"DYNNETRED_MAXPOWER"	Double	Max. power of equivalent line in MW
"DYNNETRED_POWERIGNORE"	Double	Power of machines to be ignored in MW
"DYNNETRED_PREFIX"	String	Any name prefix for reduced elements
"DYNNETRED_NODEMODELNET"	Integer	Internal display of the nodes in the network reduction 1 = PQ Type 2 = I Type 5 = PQ Type (neg. machines) 6 = I Type (neg. machines)
"DYNNETRED_NODEMODELMACHINES"	Integer	Internal display of the machines in the network reduction 1 = PQ Type 2 = I Type 3 = PV Type
"DYNNETRED_NODEMODELCOUPLING"	Integer	Internal display of coupling nodes in the network reduction 1 = PQ Type 2 = I Type 3 = PV Type 4 = S Type
"DYNNETRED_KEEPNAMES"	Integer	Name of individual machines retained 0 = no 1 = yes
"DYNNETRED_REDCONTROLLER"	Integer	Machines in the network to be reduced without controller 0 = no 1 = yes
"DYNNETRED_NOTREDCONTROLLER"	Integer	Machines in the network not to be reduced without controller 0 = no 1 = yes
"DYNNETRED_KEEPCONTROLLER"	Integer	Controller of individual machines retained 0 = no 1 = yes
"DYNNETRED_PSSCONTROLLER"	Sting	Name of the PSS controller
"DYNNETRED_EXCITER"	String	Name of the voltage controller
"DYNNETRED_GOVERNOR"	String	Name of the turbine governor

CIM Export

Parameter	Data type	Description
"CIM_FORMAT"	String	CIM version

		"CIM_V10" "CIM_V11" "CIM_V12" "CIM_V14" "CIM_V15"
"CIM_PROFILE"	String	CIM profile "CIM_STANDARD" = CIM Standard "CIM_PLANNING" = CIM for Planning "CIM_ENTSOE" = CIM for ENTSO-E
"FILENAME" "FILENAME_###"	String	Complete path and file name for the first file as well as the subsequent files. ### specifies the file number, ranging from 2 up to – at the very most – the number indicated under FILENAME_CNT.
"FILENAME_FLAG" "FILENAME_FLAG_###"	String	File type for the first data file as well as the subsequent files, ### specifies the file number, ranging from 2 up to – at the very most – the number indicated under FILENAME_CNT. "DATA" = CIM input file "BOUNDARY" = CIM boundary file "CONFIG" = CIM configuration file
"CIM_NAME"	Integer	Determines what CIM attribute is used as the name. Valid indicators are: 0 = none 1 = cim:IdentifiedObject.Name 2 = cim:IdentifiedObject.AliasName 3 = cim:IdentifiedObject.Description
"CIM_SHORTNAME"	Integer	Determines what CIM attribute is used as the short name. Valid indicators are: 0 = none 1 = cim:IdentifiedObject.Name 2 = cim:IdentifiedObject.AliasName 3 = cim:IdentifiedObject.Description
"CIM_SPLITFILES"	Integer	Export to multiple XML files 0 = export to multiple files 1 = export to a single file
"CIM_CREATEZIP"	Integer	Create ZIP archive 0 = no 1 = yes
"CIM_MRID"	String	Type of ID used "SINCALID" = ID generated by PSS SINCAL "UUID" = Universal Unique ID "GUID" = Global Unique ID
"CIM_LFRESULTS"	Integer	Export load flow results 0 = do not export any results 1 = export load flow results
"EXPORT_GRAPHIC"	Integer, String	Export graphics 0, "NONE" = no graphic export 1, "AUTOMATIC" = automatic graphic export 2, "SINCAL" = simple graphic export 3, "EXTENDED" = extended graphic export
"CIM_GENERATEDNAMES"	Integer	Export generated names 0 = no export 1 = also export generated names

CIM Import

Parameter	Data type	Description
"FILENAME_CNT"	Integer	Number of file names to be imported
"FILENAME" "FILENAME_###"	String	Complete path and file name for the first file as well as the subsequent files. ### specifies the file number, ranging from 2 up to – at the very most – the number indicated under

		FILENAME_CNT.
"FILENAME_FLAG" "FILENAME_FLAG_###"	String	File type for the first data file as well as the subsequent files, ### specifies the file number, ranging from 2 up to – at the very most – the number indicated under FILENAME_CNT. "DATA" = CIM input file "BOUNDARY" = CIM boundary file "CONFIG" = CIM configuration file
"CIM_FORMAT"	String	CIM version "CIM_V10" "CIM_V11" "CIM_V12" "CIM_V14" "CIM_V15"
"CIM_PROFILE"	String	CIM profile "CIM_STANDARD" = CIM Standard "CIM_PLANNING" = CIM for Planning "CIM_ENTSOE" = CIM for ENTSO-E
"BASE_FREQUENCY"	Double	Basic frequency
"LENGTH_FACTOR"	Double	Conversion factor for entries for length
"CIM_NAME"	Integer	Determines what CIM attribute is used as the name. Valid indicators are: 0 = none 1 = cim:IdentifiedObject.Name 2 = cim:IdentifiedObject.AliasName 3 = cim:IdentifiedObject.Description
"CIM_SHORTNAME"	Integer	Determines what CIM attribute is used as the short name. Valid indicators are: 0 = none 1 = cim:IdentifiedObject.Name 2 = cim:IdentifiedObject.AliasName 3 = cim:IdentifiedObject.Description
"IMPORT_GRAPHIC"	Integer	Import graphics 0 = no 1 = yes
"GRAPHIC_MODE"	Integer	Graphics mode 0 = schematic 1 = geographical
"GRAPHIC_INDIVIDUAL_TEXT"	Integer	Individual text for network elements and node 1 = individual text 0 = no individual text
"GRAPHIC_SCALE_FACTOR"	Double	Scaling factor of the graphics
"GRAPHIC_SYMBOLSIZE"	Integer	Size of the symbol of the network elements
"GRAPHIC_OFFSETX"	Double	X-offset of the graphics
"GRAPHIC_OFFSETY"	Double	Y-offset of the graphics

PSS E Export

Parameter	Data type	Description
"EXPORT_NAME"	Integer	Export name or short name 0 = name 1 = short name
"EXPORT_NAME_KEY"	Integer	Use nodes' short names as "BUS Number" 0 = yes 1 = no
"PSSE_VERSION"	Integer	Version number Permissible are: 32 and 33

PSS E Import

Parameter	Data type	Description
"FILENAME_CNT"	Integer	Number of the data files
"FILENAME" "FILENAME_###"	String	Complete path and file name for the first data file as well as for all subsequent files. ### specifies the file number, ranging from 2 up to – at the very most – the number indicated under FILENAME_CNT.
"PSSE_VERSION"	Integer	Version number Permissible are: 29, 30, 31, 32, 33 and 0 (Auto)
"PSSE_SEQ_FILENAME"	String	Complete path and file name of the sequence file
"PSSE_MODE"	Integer	Import mode 0 = standard mode 1 = enhanced import
"BASE_FREQUENCY"	Double	Basic frequency
"LENGTH_FACTOR"	Double	Scaling factor for lengths
"ZERO_IMPEDANCE"	Integer	Import lines without impedance 0 = no 1 = yes
"ZERO_IMPEDANCE_MIN_VALUE"	Double	Minimum impedance from which lines are considered to be connections without impedance
"PSSE_REFVOLTAGE"	Double	Reference rated voltage (for nodes and elements with 0.0 kV)
"GRAPHIC_FILENAME_CNT"	Integer	Number of the graphics files
"GRAPHIC_FILENAME" "GRAPHIC_FILENAME_###"	String	Complete path and file name for the first graphics file as well as for all subsequent files. ### specifies the file number, ranging from 2 up to – at the very most – the number indicated under GRAPHIC_FILENAME_CNT.

UCTE Export

Parameter	Data type	Description
"EXPORT_NAME"	Integer	Export name or short name 0 = name 1 = short name
"EXPORT_NAME_KEY"	Integer	Use nodes' short names as "BUS Number" 0 = yes 1 = no

DVG Export

Parameter	Data type	Description
"EXPORT_NAME"	Integer	Export name or short name 0 = name 1 = short name

DVG Import

Parameter	Data type	Description
"FILENAME"	String	Complete path and file name of the data file
"DVG_IMPORT_MODE"	Integer	Import mode (behavior during problems and errors) 0 = strict mode, abort if there are errors 1 = error-tolerant import, problems and errors are documented

"GRAPHIC_FILENAME_CNT"	Integer	Number of the graphics files
"GRAPHIC_FILENAME" "GRAPHIC_FILENAME_###"	String	Complete path and file name for the first graphics file as well as for all subsequent files. ### specifies the file number, ranging from 2 up to – at the very most – the number indicated under GRAPHIC_FILENAME_CNT.

Example

```
' Set ShortName as default for object access.
SimulateObj.Parameter( "Sim.Identification" ) = "ShortName"
' Set Name as default for object access.
SimulateObj.Parameter( "Sim.Identification" ) = "Name"
```

DoCommand – Execute Commands

Performs a predefined command in the calculations.

```
SimulateObj.DoCommand strCommand, vtParameter1, vtParameter2
```

Parameters

strCommand (String)
Predefined sign of the command to be executed.

Command	Description
"CHANGEVARIANT"	Change variant
"DELETERESULTS"	Deleting all the results in the database

vtParameter1, vtParameter2 (Variant)
Additional parameter, depending on the command.

"CHANGEVARIANT" – Change Variant

Parameter	Data type	Description
vtParameter1	String or Long Integer	Name of the variant or DB-ID of the variant
vtParameter2	Boolean	Change variant of the Include networks

"DELETERESULTS" – Deleting all the Results in the Database

Parameter	Data type	Description
vtParameter1	String or Long Integer	Name of the variant or DB-ID of the variant
vtParameter2	Boolean	Change variant of the Include networks

Example

```
' Change variant to Base.
SimulateObj.DoCommand "CHANGEVARIANT", "Base", False
```

Start – Start Calculation

Starts the calculations.

```
SimulateObj.Start strMethod
```

Parameters

strMethod (String)
Predefined sign for the calculation method.
The following table lists the parameters that can be used to specify the calculation method.

Calc. method	Description
Electrical networks	
LF	Load Flow according to Settings for Calculation Parameters
LF_NR	Load Flow Newton Raphson
LF_YMAT	Load Flow Admittance Matrix
LF_CI	Load Flow Current Iteration
LF_USYM	Unbalanced Load Flow
LF_NETO	Load Flow Netomac
LF_PSSE	Load Flow PSS E
LF_MALF	Malfunction of Selected Network Elements
LF_TRIM	Load Assignment
LF_ALLOC	Load Allocation
LF_BAL	Load Balancing
LF_RESUP	Restoration of Supply
LF_TAP	Tap Zone Detection
LF_INC	Load Development
LC	Load Profile
GEN_PV	PV Curves
OPT_LF	Load Flow Optimization
OPT_BR	Optimal Branching
OPT_COMP	Compensation Power
OPT_CAP	Capacitor Placement
OPT_NET	Optimal Network Structure
COND	Contingency Analysis
SC1	1-Phase Ground Fault
SC2	2-Phase Short Circuit
SC3	3-Phase Short Circuit
SC1 [NodeID]	1-Phase Ground Fault at Node
SC2 [NodeID]	2-Phase Short Circuit at Node
SC3 [NodeID]	3-Phase Short Circuit at Node
IC[x][R][E] [NodeID]	Individual Short Circuit (Number of Phases, Return Line Fault, Ground Fault) at Node
GC2	2-Phase Ground Fault
MF	Multiple Faults
DIM	Low-Voltage Dimensioning
HAR	Harmonics
RC	Ripple Control
ECO_SUM	Economic Efficiency

MOT	Motor Start-Up
MOT_SIMPLE	Simple Motor Start-Up
NETO_STAB	Stability
NETO_TSTAB	Transient Stability
NETO_EW	Eigenvalues
REL	Reliability
REL_EVAL	Reliability Evaluations
PROT SC1 [FaultID]	Protection Coordination – 1-Phase Ground Fault
PROT SC2 [FaultID]	Protection Coordination – 2-Phase Short Circuit
PROT GC2 [FaultID]	Protection Coordination – 2-Phase Ground Fault
PROT SC3 [FaultID]	Protection Coordination – 3-Phase Short Circuit
PROT IC[x][R][E] [FaultID]	Protection Coordination – Individual Short Circuit (Number of Phases, Return Line Fault, Ground Fault)
PROT MF	Fault Summary for Multiple Faults
PROT_DET	Determining Fault Locations
PROT_SET	DI Device – Settings
PROT_SET_CHART	DI Device – Charts
PROT_ROUTE SC1	Route – 1-Phase Ground Fault
PROT_ROUTE SC2	Route – 2-Phase Ground Fault
PROT_ROUTE GC2	Route – 2-Phase Short Circuit
PROT_ROUTE SC3	Route – 3-Phase Short Circuit
PROT_ROUTE IC[x][R][E]	Protection Routes – Individual Short Circuit (Number of Phases, Return Line Fault, Ground Fault)
ARCFL	Arc Flash
ZUBER	Reliability
ZUBER_EVAL	Reliability Evaluations
Water networks	
FLOW_H2O	Steady-State
FLOW_H2O_TM	Time Series
FLOW_H2O_OP	Operating Series
FLOW_H2O_COND	Contingency Analysis
FLOW_H2O_MALF	Contingency Analysis (Selected Elements)
FLOW_H2O_FWP	Fire Water Pressure
FLOW_H2O_FWQ	Fire Water Amount
FLOW_H2O_LEAKP	Fire Water Pressure (Selected Elements)
FLOW_H2O_LEAKQ	Fire Water Amount (Selected Elements)
Gas networks	
FLOW_GAS	Steady-State
FLOW_GAS_TM	Time Series
FLOW_GAS_OP	Operating Series
FLOW_GAS_COND	Contingency Analysis
FLOW_GAS_MALF	Contingency Analysis (Selected Elements)
Heating/cooling networks	
FLOW_HEAT	Steady-State
FLOW_HEAT _TM	Time Series
FLOW_HEAT _OP	Operating Series
FLOW_HEAT_COND	Contingency Analysis
FLOW_HEAT_MALF	Contingency Analysis (Selected Elements)

The following table lists the parameters that can be used to specify the import and export functions.

Import/export	Description
Electrical networks	
CIM_IMP	CIM Import
CIM_EXP	CIM Export
PSSE_IMP	PSS E Import
PSSE_EXP	PSS E Export
UCTE_IMP	UCTE Import
UCTE_EXP	UCTE Export
DVG_IMP	DVG Import
DVG_EXP	DVG Export

Example

```
' Start load flow simulation.
SimulateObj.Start "LF NR"
If SimulateObj.StatusID = 1101 Then
    WScript.Echo "Simulation finished without errors!"
Else
    WScript.Echo "Error: Load flow failed!"
    Exit Do
End If
```

StatusID – Status Code of Calculation Procedure

Queries the status of the calculation procedure.

```
lStatus = SimulateObj.StatusID
```

Properties

StatusID (Long Integer)
Status code for the calculation procedure. "1101" indicates that the calculations were error-free.

Example

```
' Check if the simulation was finished without any errors.
If Not ( SimulateObj.StatusID = 1101 ) Then
    WScript.Echo "Error: Simulation failed!"
End If
```

GetObj – Access Calculation Objects

Returns an instance for a calculation object that lets you directly access "internal" objects created in the calculations.

```
Set LoadObj = SimulateObj.GetObj( strObjectType, strName )
Set LoadObj = SimulateObj.GetObj( strObjectType, lDBID )
```

Parameters

strObjectType (String)
Object type of the network element. This is the name of the database table where the element data are stored.

lDBID (String)
Database ID of the network element.

strName (String)
Name of the network element. A global parameter can be assigned to use the name or the short name for identification.

Return Value

Object (Object)
Automation object of a network element created in the calculations.

Example

```
' Get SimulationObject of type "LOAD" with name "LO8".
Dim LoadObj
Set LoadObj = SimulateObj.GetObj( "LOAD", CStr( "LO8" ) )
If LoadObj is Nothing Then
    WScript.Echo "Error: Load not found!"
    WScript.Quit
End If
```

GetObjById, GetObjByGUID – Access Calculation Objects Using the ID

Returns an instance for a calculation object that lets you directly access "internal" objects created in the calculations.

```
Set SimObj = SimulateObj.GetObjByID( lID )
Set SimObj = SimulateObj.GetObjByGUID( strGUID )
```

Parameters

lID (Long Integer)
Internal number of the calculation object. See the topology data of the calculation objects for this "Internal ID".

strGUID (String)
Master Resource of the calculation object.

Return Value

SimObj (Object)
Automation object of a network element created in the calculations.

Example

```
' Get SimulationObject with internal ID 8.
Dim SimObj
Set SimObj = SimulateObj.GetObjByID( 8 )
If SimObj Is Nothing Then
    WScript.Echo "Error: Object not found!"
    WScript.Quit
End If

' Get SimulationObject with Master Resource ID.
Dim SimObj
Set SimObj = SimulateObj.GetObjByGUID( "289DAFC1-8541-4abb-AE9F-1C47E6A2D32B" )
If SimObj Is Nothing Then
    WScript.Echo "Error: Object not found!"
    WScript.Quit
End If
```

DB_EL, DB_FLOW – Access Database Objects

Lets you access the input data and results of the simulation.

```
Set SimulateDatabase = SimulateObj.DB_EL
Set SimulateDatabase = SimulateObj.DB_FLOW
```

Properties

DB_EL (Object)
Database object for electrical networks.

DB_FLOW (Object)
Database object for pipe networks.

Comments

This property is read-only.

Example

```
' Get the database object.
Dim SimulateDatabase
Set SimulateDatabase = SimulateObj.DB_EL
If SimulateDatabase Is Nothing Then
    WScript.Echo "Error: Getting database object failed!"
    WScript.Quit
End If
' ...
Set SimulateDatabase = Nothing
```

Messages – Access Message Objects

Lets you access messages generated by the calculations.

```
Set objMessages = SimulateObj.Messages
```

Properties

Messages (Object)
Automation object for the calculation messages.

Comments

This property is read-only.

Example

```
' Get message object.
Dim objMessages
Set objMessages = SimulateObj.Messages

' Release message object.
Set objMessages = Nothing
```

4.1.2 Calculation Object

This object directly accesses "internal" objects in the calculations that describe the network model. This object represents the network element display in the main memory of the calculations.

The calculation object lets you directly manipulate input data in the calculations. For example, you can modify values for a load's active and reactive power without loading data from the database. The calculation object even has functions to simplify accessing the results of the network elements.

For a list of attributes addressed by the calculation object, see the chapter on **Attributes of Calculation Objects**.

Simulation objects with the **GetObj** function create an instance for a calculation object.

Example

```
' Get simulation object of type "Load" with name "LO8".
Dim LoadObj
Set LoadObj = SimulateObj.GetObj( "LOAD", "LO8" )
If LoadObj Is Nothing Then
    WScript.Echo "Error: Load not found!"
    WScript.Quit
End If

' Release the simulation object.
Set LoadObj = Nothing
```

Count – Number of Possible Attributes

Returns the number of the possible attributes for the respective object.

```
lCnt = LoadObj.Count()
```

Return Value

lCnt (Long Integer)
Number of possible attributes.

Example

```
' Get the number of available attributes.
Dim Cnt
Cnt = LoadObj.Count()
```

Name – Determine Attribute Names

Returns the name of an attribute.

```
strName = SimObj.Name( iAttribute )
```

Properties

Name (String)
Name of the attribute.

Parameters

iAttribute (Long Integer)
Number of the attribute.

Example

```
' Display the names of all available attributes for the object.
Dim iAttr, lCnt
lCnt = SimObj.Count()
For iAttr = 1 To lCnt
    Dim strName
    strName = SimObjObj.Name( iAttr )
    WScript.Echo iAttr & ": " & strName
Next
```

Item – Access Attributes

Lets you access the different input and output data for a calculation object.

```
Value = SimObj.Item( lAttribute )
Value = SimObj.Item( strAttribute )

SimObj.Item( lAttribute ) = Value
SimObj.Item( strAttribute ) = Value
```

Properties

Item (Variant)
Value of the attribute.

Parameters

iAttribute (Long Integer)
Numerical index of the attribute.

strAttribute (Long Integer)
Name of the attribute.

Comments

What attributes are actually available depends on the object. All objects have identical topology attributes. These topology attributes uniquely identify nodes and network elements and are used to switch them ON and OFF.

For a detailed presentation of all the objects and their available attributes, see the chapter **Calculation Objects and their Attributes**.

Example

```
' Get P from the object and set a new value for this attribute.
Dim Val
Val = SimObj.Item( "P" )
SimObj.Item( "P" ) = P * 2

' Get the value with the index 3 and assign a new value.
Dim Val
Val = SimObj.Item( 3 )
SimObj.Item( 3 ) = Val * 2
```

Result – Access Calculation Results Object

Lets you access the results object of a calculations object.

```
Set ResultObj = SimObj.Result( strResult, iTerminalNo )
```

Parameters

strResult (String)
SQL name of the desired results table.

iTerminalNo (Integer)
Connection number for the desired results.

Return Value

ResultObj (Object)
Automation object for the results.

Comments

PSS SINCAL provides network element results for each terminal. Node elements (such as generators, asynchronous machines and loads) have one terminal and branch elements (such as lines, transformers and serial reactors) have two terminals.

Example

```
' Get load flow results on the first terminal of object.
Dim ResultObj
Set ResultObj = SimObj.Result( "LFBRANCHRESULT", 1 )

If ResultObj Is Nothing Then
    WScript.Echo "Error: No result available!"
End If
```

4.1.3 Calculation Results Object

Calculation results objects are virtual objects that simplify accessing results for individual network elements.

Calculation objects with the **Result** function produce an instance of a calculation results object.

Example

```
' Get the load flow result object for a load.
Dim LFBranchResultLoad
Set LFBranchResultLoad = LoadObj.Result( "LFBRANCHRESULT", 1 )

If LFBranchResultLoad Is Nothing Then
    WScript.Echo "Error: Cant get result object!"
End If
```

If the instance of the automation object is not used any longer, it must be released with the following instruction.

```
' Relase the result object.
Set LFBranchResultLoad = Nothing
```

Count – Number of Possible Attributes

Returns the number of the possible attributes for the respective results object.

```
lCnt = ResultObj.Count()
```

Return Value

lCnt (Long Integer)
Number of possible attributes.

Example

```
' Get the number of available attributes.
Dim Cnt
Cnt = ResultObj.Count()
```

Name – Determine Attribute Names

Returns the name of an attribute.

```
strName = ResultObj.Name( lAttribute )
```

Properties

Name (String)
Name of the attribute.

Parameters

lAttribute (Long Integer)
Number of the attribute.

Example

```
' Display the names of all available attributes for the object.
Dim iAttr, lCnt
lCnt = ResultObj.Count()
For iAttr = 1 To lCnt
    Dim strName
    strName = ResultObj.Name( iAttr )
    WScript.Echo iAttr & ": " & strName
Next
```

Item – Access Attributes

Lets you access the output data for a results object.

```
Value = ResultObj.Item( lAttribute )
Value = ResultObj.Item( strAttribute )
```

Properties

Item (Variant)
Value of the attribute.

Parameters

iAttribute (Long Integer)
Numerical index of the attribute.

strAttribute (Long Integer)
Name of the attribute.

Comments

This property is read-only.

The calculation results object is exactly the same as the results tables in the PSS SINCAL database. This means the attribute names are the same as the field names in the results table. For detailed documentation of all results tables, see the **PSS SINCAL Database Description**.

Example

```
' Getting load flow result for node.
Dim LFNodeResult
Set LFNodeResult = NodeObj.Result( "LFNodeResult", 0 )
If LFNodeResult Is Nothing Then
Else
    Dim u un
    u un = LFNodeResult.Item( "U Un" )
    WScript.Echo "Node voltage at node U/Un = " & u un & "%"
    Set LFNodeResult = Nothing
End If
```

4.1.4 Database Object

This object lets you access the individual database tables.

Example

```
' Database object.
Dim SimulateDatabase
Set SimulateDatabase = SimulateObj.DB EL
If SimulateDatabase Is Nothing Then
    WScript.Echo "Error: Getting database object failed!"
    WScript.Quit
End If
```

If the instance of the automation object is not used any longer, it must be released with the following instruction.

```
' Release the tabular object.
Set SimulateDataBase = Nothing
```

GetRowObj – Determine Instance for a Tabular Object

Provides an instance for a tabular object.

```
Set TableObj = SimulateDatabase.GetRowObj( strTable )
```

Parameters

strTable (String)
Name of the database table.

Return Value

TableObj (Object)
Automation object of the database table.

Example

```
' Get the load flow node result tabular object.
Dim LFNodeResult
Set LFNodeResult = SimulateDatabase.GetRowObj( "LFNodeResult" )
If LFNodeResult Is Nothing Then
    WScript.Echo "Error: Getting LFNodeResult object failed!"
    WScript.Quit
End If
```

4.1.5 Tabular Object

This object contains all the data from a database table. The data in the tabular object are organized in lines and columns, similar to a calculation table. Each line represents a unique record and each column a field or attribute within the record.

The database object creates an instance of a tabular object.

Example

```
' Get the load flow node result tabular object.
Dim LFNodeResult
Set LFNodeResult = SimulateDataBase.GetRowObj( "LFNodeResult" )
If LFNodeResult Is Nothing Then
    WScript.Echo "Error: Getting LFNodeResult object failed!"
    WScript.Quit
End If
```

If the instance of the automation object is not used any longer, it must be released with the following instruction.

```
' Release the tabular object.
Set LFNodeResult = Nothing
```

Open – Open a Tabular Object

Opens the database table for a tabular object so you can access the records.

```
hr = TableObj.Open()
```

Return Value

hr (HRESULT)
Status code of the command.

Example

```
' Open a database table.
Dim hr
hr = TableObj.Open()
If hr <> 0 Then
    WScript.Echo "Error: Open failed!"
    WScript.Quit
End If
```

Close – Close a Tabular Object

Closes a database table of a tabular object.

```
TableObj.Close
```

Comments

Once you close the database table, you must not access the tabular object any longer. The instance of the tabular object also needs to be released.

Example

```
' Close a database table and release it.
TableObj.Close
Set LFNodeResult = Nothing
```

CountRows – Determine the Number of Records

Provides the number of the records in a tabular object.

```
lCnt = TableObj.CountRows
```

Properties

CountRows (Long Integer)
Number of the records in a tabular object.

Example

```
' Open a database table and determine the count of records in it.
Dim hr
hr = TableObj.Open()
If hr <> 0 Then
    WScript.Echo "RecordCount: " & TableObj.CountRows
End If
```

MoveFirst, MoveLast, MoveNext, MovePrev – Position in the Data

Positions the data cursor inside the data.

```
hr = TableObj.MoveFirst()
hr = TableObj.MoveNext()
hr = TableObj.MoveLast()
hr = TableObj.MovePrev()
```

Return Value

hr (HRESULT)
Status code of the command.

Comments

MoveFirst positions the data cursor at the beginning of the table, **MoveLast** positions it at the last record in a table. **MoveNext** and **MovePrev** move the data cursor to the next or previous record.

Example

```
' Move to the first record and start reading.
Dim hr
hr = LFNodeResult.MoveFirst()
Do While hr = 0

    ' Your own code is here ...

    ' Move to next records.
    hr = LFNodeResult.MoveNext()
Loop
```

Count – Number of Possible Attributes

Returns the number of the attributes for the tabular object.

```
lCnt = TableObj.Count
```

Return Value

lCnt (Long Integer)
Number of possible attributes.

Comments

This property is read-only.

Example

```
' Get the number of available attributes.
Dim lCnt
lCnt = TableObj.Count
```

Name – Determine Attribute Names

Returns the name of an attribute.

```
strName = TableObj.Name( iAttribute )
```

Properties

Name (String)
Name of the attribute.

Parameters

iAttribute (Long Integer)
Number of the attribute.

Example

```
' Display the names of all available attributes for the object.
Dim iAttr, lCnt
lCnt = TableObj.Count
For iAttr = 1 To lCnt
    Dim strName
    strName = TableObj.Name( iAttr )
    WScript.Echo iAttr & ": " & strName
Next
```

Item – Access Attributes

Provides access to the individual attributes (fields) for the current line of data.

```
Value = TableObj.Item( lAttribute )
Value = TableObj.Item( strAttribute )
```

Properties

Item (Variant)
Value of the attribute.

Parameters

iAttribute (Long Integer)
Numerical index of the attribute.

strAttribute (Long Integer)
Name of the attribute.

Example

```
' Get U_Un from the table.
Dim Val
Val = TableObj.Item( "U_Un" )
```

4.1.6 Message Object

The message object accesses the messages generated during the calculations.

Use the simulation object to access the message object. The following example shows how to display all the calculation messages with a program loop.

Example

```
' Get messages from simulation.
Dim objMessages
Set objMessages = SimulateObj.Messages

Dim strType
Dim intMsgIdx
For intMsgIdx = 1 To objMessages.Count
    Dim Msg
    Set Msg = objMessages.Item( intMsgIdx )

    Select Case Msg.Type
        case 1 ' STATUS
        case 2 ' INFO
        case 3 ' WARNING
            WScript.Echo Msg.Text
        case 4 ' ERROR
            WScript.Echo Msg.Text
    End Select

    Set Msg = Nothing
Next

' Release message object if not longer needed.
Set objMessages = Nothing
```

Count – Number of Possible Messages

Returns the number of the messages.

```
Cnt = objMessages.Count
```

Properties

Count (Long Integer)
Number of available messages.

Comments

This property is read-only.

Example

```
' Get the number of available messages.
Dim Cnt
Cnt = objMessages.Count
WScript.Echo "Number of Messages: " & intMsgCnt
```

Item – Access Message Data Object

Provides access to a message data object.

```
Set Msg = objMessages.Item( iMsgIndex )
```

Properties

Item (Object)
Automation object of a message.

Parameters

iMsgIndex (Long Integer)
Numerical index of the message. Permitted indices start with "1" and end with the number of messages.

Comments

This property is read-only.

Example

```
' Get a message.
Dim Msg
Set Msg = objMessages.Item( 1 )
```

4.1.7 Message Data Object

This object represents a message used to call up the message text, the message type and other data for the message.

Example

```
' Get the first message and display the message type and text.
Dim Msg
Set Msg = objMessages.Item( 1 )

Select Case Msg.Type
    case 1 ' STATUS
       WScript.Echo "Status: " & Msg.Text
    case 2 ' INFO
       WScript.Echo "Info: " & Msg.Text
    case 3 ' WARNING
       WScript.Echo "Warning: " & Msg.Text
```

```
    case 4 ' ERROR
       WScript.Echo "Error: " & Msg.Text
End Select
```

Text – Message Text

Lets you access the text of a calculation message.

```
strText = Msg.Text
```

Properties

Text (String)
Message text.

Comments

This property is read-only.

Example

```
' Get the first message and display the message text.
Dim Msg
Set Msg = objMessages.Item( 1 )

WScript.Echo Msg.Text
```

Type – Message Type

Specifies the type of the message.

```
iType = Msg.Type
```

Properties

Type (Integer)
Predefined code for the type of message.

Code	Message type	Description
1	Status	The calculation methods display status messages while they perform different functions.
2	Info	Information messages contain general information (number of isolated nodes, etc.).
3	Warning	Warnings show tolerable errors in the calculations.
4	Error	Error messages show serious errors in the calculations and indicate that the calculations could not be completed properly.

Example

```
' Get the first message and display the message type.
Dim Msg
Set Msg = objMessages.Item( 1 )
```

```
Select Case Msg.Type
    case 1 ' STATUS
       WScript.Echo "Status"
    case 2 ' INFO
       WScript.Echo "Info"
    case 3 ' WARNING
       WScript.Echo "Warning"
    case 4 ' ERROR
       WScript.Echo "Error"
End Select
```

CountObjectIds – Number of Network Elements

Specifies how many network elements the message refers to.

```
lCntElements = Msg.CountObjectIds
```

Properties

CountObjectIds (Long Integer)
Number of network elements.

Comments

This property is read-only.

Example

```
' Loop over all objects of the current message.
If Msg.CountObjectIds > 0 Then
    Dim i
    For i = 1 To Msg.CountObjectIds
        ' ...
    Next
End If
```

ObjectIdAt, ObjectTypeAt – Network Element Data

Provides the Object ID and the object type for a network element.

```
lObjID   = Msg.ObjectIdAt( lIndex )
iObjType = Msg.ObjectTypeAt( lIndex )
```

Parameters

lIndex (Long Integer)
Numerical index starting with 1.

Properties

ObjectIdAt (Long Integer)
Network element ID.

ObjectTypeAt (Integer)
Database and type of the network element in a screen form.

Comments

The object type is stored in the first 12 bits and the database type in the highest 4 bits. To determine the object type, the bit mask has to be combined with 0FFF. You get the database type by moving the bits 12 spaces to the right.

These properties are read-only.

Example

```
' Loop over all objects of the current message.
If Msg.CountObjectIds > 0 Then
    Dim i
    Dim strObjects
    For i = 1 To Msg.CountObjectIds
        If strObjects <> "" Then
            strObjects = strObjects & ", "
        End If

       ' Determine database and object type.
        Dim sDBType
        Dim sRowType
        sDBType = Msg.ObjectTypeAt( i ) \ &H1000
        sRowType = Msg.ObjectTypeAt( i ) And &H0FFF

        strObjects = strObjects & Msg.ObjectIdAt( i ) & "(" & sRowType & ")"
    Next
    WScript.Echo strObjects
End If
```

4.2 Calculation Objects and their Attributes

4.2.1 Available Calculation Objects

The following tables show the calculation objects that are available for the different network types.

Electrical Networks

Object type	Description
General Data	
CalcParameter	Calculation settings
VoltageLevel	Network level
NetworkGroup	Network area

Nodes/Busbars	
Node	Node
Node Elements	
SynchronousMachine	Synchronous machine
PowerUnit	Power unit
Infeeder	Infeeder
DCInfeeder	DC-Infeeder
AsynchronousMachine	Asynchronous machine
Load	Load
ShuntImpedance	Shunt impedance
ShuntReactor	Shunt reactor
ShuntCondensator	Shunt capacitor
VarShuntElement	Variable serial element
HarResNet	Shunt Harmonics Resonance Network
Branch Elements	
TwoWindingTransformer	Two-winding transformer
ThreeWindingTransformer	Three-winding transformer
Line	Line
VarSerialElement	Variable shunt element
SerialReactor	Serial reactor
SerialCondensator	Serial capacitor
HarBranchResNet	Serial Harmonics Resonance Network
Additional Elements	
ProtOCFault	Fault observation
ProtLocation	Protection location

Water Networks

Object type	Description
General Data	
FlowCalcParameter	Calculation settings
FlowNetworkLevel	Network level
FlowNetworkGroup	Network area
Nodes/Busbars	
FlowNode	Node
Node Elements	
FlowWaterTower	Water tower
FlowPump	Pump
FlowConsumer	Consumer
FlowPressureBuffer	Pressure buffer
FlowLeakage	Leakage
Branch Elements	
FlowLine	Line
FlowValve	Sliding valve/non-return valve

FlowPumpLine	Pressure increase pump
FlowConstLine	Const. pressure decrease/const. flow
FlowPressureReg	Pressure regulator

Gas Networks

Object type	Description
General Data	
FlowCalcParameter	Calculation settings
FlowNetworkLevel	Network level
FlowNetworkGroup	Network area
Nodes/Busbars	
FlowNode	Node
Node Elements	
FlowInfeederG	Infeeder gas
FlowConsumer	Consumer
FlowPressureBuffer	Pressure buffer
FlowLeakage	Leakage
Branch Elements	
FlowLine	Line
FlowValve	Sliding valve/non-return valve
FlowConstLine	Const. pressure decrease/const. flow
FlowPressureReg	Pressure regulator
FlowCompressor	Compressor

Heating/Cooling Networks

Object type	Description
General Data	
FlowCalcParameter	Calculation settings
FlowNetworkLevel	Network level
FlowNetworkGroup	Network area
Nodes/Busbars	
FlowNode	Node
Node Elements	
FlowInfeederH	Infeeder heating/cooling
FlowPump	Pump
FlowConsumer	Consumer
FlowPressureBuffer	Pressure buffer
FlowLeakage	Leakage

Branch Elements	
FlowLine	Line
FlowValve	Sliding valve/non-return valve
FlowPumpLine	Pressure increase pump
FlowConstLine	Const. pressure decrease/const. flow
FlowPressureReg	Pressure regulator
FlowThermoReg	Temperature regulator
FlowHeatExchanger	Heat Exchanger

4.2.2 General Topology Attributes

General topology attributes are available for both nodes and network elements. They let you query important basic information such as names and connection phases as well as establishment and shutdown times.

Attribute	State	Description
Node		
TOPO.ID	Read	Internal ID – Node
TOPO.DBID	Read	Database ID – Node (Node_ID)
TOPO.Name	Read	Name – Node
TOPO.ShortName	Read	Short Name – Node
TOPO.Phase	Read	Connection Phase (is determined dynamically by the attached elements) 1: L1 2: L2 3: L3 4: L12 5: L23 6: L31 7: L123
TOPO.TI	Read / Write	Establishment Time
TOPO.TS	Read / Write	Shutdown Time
Network Elements		
TOPO.ID	Read	Internal ID – Network Element
TOPO.DBID	Read	Database ID – Network Element (Element_ID)
TOPO.Name	Read	Name – Network Element
TOPO.ShortName	Read	Short Name – Network Element
TOPO.State	Read / Write	Operational Status – Network Element 0: Out of service 1: In service
TOPO.TI	Read / Write	Establishment Time
TOPO.TS	Read / Write	Shutdown Time
TOPO.Node1.ID	Read	Internal ID – 1. Node (up to 3 nodes are possible)
TOPO.Node1.DBID	Read	Database ID – 1. Node
TOPO.Terminal1.ID	Read	Internal ID – 1. Terminal (up to 3 terminals are possible)
TOPO.Terminal1.DBID	Read	Database ID – 1. Terminal
TOPO.Terminal1.State	Read / Write	Switching Status – 1. Terminal 0: Switches opened 1: Switches closed
TOPO.Terminal1.Phase	Read	Connection Phase 1: L1 2: L2

		3: L3 4: L12 5: L23 6: L31 7: L123
Additional Elements		
TOPO.ID	Read	Internal ID – Additional Element
TOPO.DBID	Read	Database ID – Additional Element (Element_ID)
TOPO.Name	Read	Name – Additional Element

The **ID** attribute has a unique key for the unique identification of every object in the calculation methods.

DBID has the Database ID of the specific node, network element or terminal.

The **State** attribute is a special feature for network elements and their terminals. This attribute shows the operating status of the network element or the tripping status of the respective terminal. Simply changing this attribute can switch network elements ON or OFF.

4.2.3 Attributes of Calculation Objects for Electrical Networks

Calculation Settings (CalcParameter)

Attribute name	Data type	Unit	Description
ViewDate	Double		View Date
LoadDataDate	Double		Load Data Date
Sref	Double	MVA	Reference Power
FrqNet	Double	Hz	Frequency
Uref	Double	kV	Reference Voltage
ExportForm	Integer		Export Format for Names 0: Name 1: Short name
IncreaseLoads	Integer		Use Increased Loads 0: No 1: Yes
ContrAdjustment	Integer		Controller Adjustment 1: Discrete 2: Continuous
FlatStart	Integer		Flat Start 0: No 1: Yes
ChangeLFMethode	Integer		Change Load Flow Method at Convergence Problems 0: Off 1: On
LFPreCalc	Integer		Pre-Calculate 0: No 1: Yes
LFMethod	Integer		Load Flow Procedure 1: Current iteration 2: Newton-Raphson 3: Admittance matrix 5: Unbalanced

StoreRes	Integer		Include Load Profile Result in Database 0: Due to method 1: All 2: Restricted elements only 3: All elements in case of restrictions
ImpLoad	Integer		Impedance Load Conversion 0: No 1: Normal 2: Extended
LFControl	Integer		Enable Automatic Controller Change 0: No 1: Normal 2: Extended
Island	Integer		Island Operation 0: No 1: Yes
StartTime	Double		Start Time Load Profile
Duration	Double		Duration Load Profile
TimeStep	Double		Time Step Load Profile
IncrStartDate	Double		Start Date Load Increase
IncrEndDate	Double		End Date Load Increase
PeakCurrentCalc	Integer		Peak Short Circuit Current Calculation Type 1: Meshed network 2: Non-meshed network 3: Equivalent frequency procedure
TrippCurrentCalc	Integer		Tripping Current Calculation Type 1: IANEU VDE0102/1.90 – IEC 909 2: IAALT VDE0102/10.71
EcoInflation	Double	%	Cost increase

Network Level (VoltageLevel)

Attribute name	Data type	Unit	Description
Flag_Volt	Integer		Kind of Voltage 1: Line – Line Voltage 2: Line – Earth Voltage
Un	Double	kV	Nominal Voltage
Uop	Double	kV	Network Operating Voltage
f	Double	Hz	Frequency
fRD	Double	Hz	Ripple Control Frequency
Temp_Line	Double	°C	Overhead Line Conductor Temperature
Temp_Cabel	Double	°C	Cable Conductor Temperature
AmbTemp_Cable	Double	°C	Ambient Temperature of Cable
AmbTemp_Line	Double	°C	Ambient Temperature of Overhead Line
TempSun_Line	Double	°C	Temperature Increase due to Sun at Overhead Line
CalcSC	Integer		Calculate Short Circuit 0: No 1: Yes
CalcNpt	Integer		Calculate Current through Neutral Points 0: No 1: Yes
FlagUsc	Integer		Voltage Data due to VDE/IEC 1: c-value 2: Source voltage
Uk	Double	kV	Source Voltage
c	Double	1	c Value
ts	Double	s	Switch Delay

Ipmax	Double	kA	Maximum Admissible Surge Current
Ibrkmax	Double	kA	Maximum Admissible Tripping Current
Upre	Double	pu	Pre-Fault Voltage due to ANSI/IEEE
Flag_Toleranz	Integer		Voltage Tolerance – Low Voltage Networks 1: 6 % 2: 10 %

Network Area (NetworkGroup)

Attribute name	Data type	Unit	Description
Flag_IC	Integer		Transfer Active 0: No 1: Yes
Pdes	Double	MW	Interchange Leaving the Area
Ptol	Double	MW	Interchange Tolerance Bandwidth
Flag_Malfunc	Integer		Malfunction 0: None 1: All elements 2: Loaded elements 3: All lines 4: Loaded lines 5: All lines and transformers 6: Loaded lines and transformers
Flag_Connectors	Integer		Consider Connectors in Malfunction and Caused Malfunction 0: No 1: Yes
Util_BaseLimit	Double	%	Base Utilization Limit
Flag_CausedMalfunc	Integer		Caused Malfunction 0: None 1: Marked areas 2: Own area
Flag_CausedElem	Integer		Caused Elements 1: Loaded elements 2: Loaded lines 3: Loaded lines and transformers
Util_CausedLimit	Double	%	Caused Utilization Limit
Flag_CausedForeign	Integer		Marked for Caused Malfunction 0: No 1: Yes
Flag_Util	Integer		Show Elements outside Limits 0: None 1: Elements and nodes 2: Elements 3: Lines, transformers and nodes 4: Lines and transformers 5: Lines and nodes 6: Lines

Node (Node)

Attribute name	Data type	Unit	Description
Uref	Double	kV	Voltage Target Value
uul	Double	%	Voltage Upper Limit
ull	Double	%	Voltage Lower Limit
StartU	Double	kV	Initial Voltage
StartPhi	Double	°	Angle – Initial Voltage

Ik2	Double	kA	Maximum Admissible Short Circuit Current
Ip	Double	kA	Maximum Admissible Peak Short Circuit Current
uul1	Double	%	Additional Voltage Upper Limit
ull1	Double	%	Additional Voltage Lower Limit
FlagPhase	Integer		Preferred Fault Phase 1: L1 2: L2 3: L3 4: L12 5: L23 6: L31 7: L123

Synchronous Machine (SynchronousMachine)

Attribute name	Data type	Unit	Description
Flag_Machine	Integer		Type of Machine 1: Turbo generator 2: Hydro gen. (amort.) 3: Hydro generator 4: Condenser 5: Non-interconnected equivalent 6: Power station equivalent 7: Transmission system equivalent 8: Distribution system equivalent
Sn	Double	MVA	Rated Apparent Power
P	Double	MW	Active Power
Q	Double	Mvar	Reactive Power
u	Double	%	Generator Voltage Percentage
Un	Double	kV	Rated Voltage
Ug	Double	kV	Generator Voltage Absolute
R_X	Double	pu	Ratio R/X – Positive-Phase Sequence
xd2sat	Double	%	Saturated Subtransient Reactance
xd1sat	Double	%	Saturated Transient Reactance
xi	Double	%	Internal Reactance
Ugmax	Double	%	Maximum Generator Voltage
Ikp	Double	kA	Sustained Short Circuit Current of Compound Machines
Flag_LF	Integer		Load Flow Type 1: |I| and phi 2: P and Q 3: |usrc| and delta 4: |S| and cosphi 5: |Usrc| and delta 6: P and |u| 7: P and |U| 8: |uterm| and delta 9: |Uterm| and delta
I	Double	kA	Basic Current Source
delta	Double	°	Voltage Angle
S	Double	MVA	Apparent Power
cosphi	Double	1	Power Factor
Zero- and Negative-Phase Sequence			
Flag_Z0	Integer		Grounding 0: Not grounded 1: Fixed grounded 2: Grounded w. impedances

Flag_Z0Input	Integer		Zero-Phase Sequence Input Data 1: Z0/Z1 and R0/X0 2: R0 and X0
R0_X0	Double	pu	Ratio R/X – Zero-Phase Sequence
Z0_Z1	Double	pu	Ratio Zero-Phase to Positive-Phase Sequence Impedance
R0	Double	Ohm	Resistance – Zero-Phase Sequence
X0	Double	Ohm	Reactance – Zero-Phase Sequence
X22	Double	%	Reactance – Negative-Phase Sequence
R2_X2	Double	pu	Ratio R/X – Negative-Phase Sequence
Short Circuit Data			
Flag_Tau	Double		Current Decay 0: No 1: Yes
Decay_Tau	Double	s	Decay Time Constant
Protection Data			
Flag_Decay	Double		State Decay Current 0: No Data 1: Ia 2: Iasym
Decay_Uwrk	Double	pu	Decay Working Voltage
Controller Data			
Unode	Double	%	Controlled Voltage at Controller Node
Control Range Data			
Pmin	Double	MW	Active Power – Lower Limit
Pmax	Double	MW	Active Power – Upper Limit
Qmin	Double	Mvar	Reactive Power – Lower Limit
Qmax	Double	Mvar	Reactive Power – Upper Limit
Umin	Double	%	Voltage Lower Limit
Umax	Double	%	Voltage Upper Limit
cosphi_lim	Double		Limit Power Factor
Dynamic Data			
xd2	Double	pu	Subtransient Reactance
Reliability Data			
Flag_LP	Integer		Load Priority 1: High 2: Medium 3: Normal 4: Small 5: Low
CustCnt	Long Integer	1	Number of Supplied Customers

Power Unit (PowerUnit)

Attribute name	Data type	Unit	Description
Flag_Machine	Integer		Type of Machine 1: Turbo generator 2: Hydro gen. (amort.) 3: Hydro generator 4: Condenser 5: Non-interconnected equivalent 6: Power station equivalent 7: Transmission system equivalent 8: Distribution system equivalent
Sn	Double	MVA	Rated Apparent Power
Un	Double	kV	Rated Voltage
R_X	Double	pu	Ratio R/X – Positive-Phase Sequence
xd2	Double	%	Subtransient Reactance
xi	Double	%	Internal Reactance
Ugmax	Double	%	Maximum Generator Voltage
Ug	Double	kV	Rated Voltage Generator
cosphin	Double	1	Rated Power Factor
Ikp	Double	kA	Sustained Short Circuit Current of Compound Machines
xd1sat	Double	%	Saturated Transient Reactance
xd2sat	Double	%	Saturated Subtransient Reactance
Un2	Double	kV	Rated Voltage Transformer – Network Side
Un1	Double	kV	Rated Voltage Transformer – Generator Side
Snt	Double	MVA	Rated Apparent Power Transformer
Smax	Double	MVA	Full Load Power
ur	Double	%	Short Circuit Voltage – Ohmic Part
Flag_LF	Integer		Load Flow Type 1: |I| and phi 2: P and Q 3: |usrc| and delta 4: |S| and cosphi 5: P and |u| 6: |Usrc| and delta 7: P and |U| 8: |uterm| and delta 9: |Uterm| and delta
phi	Double	°	Phase Angle
I	Double	kA	Basic Current Source
P	Double	MW	Active Power
Q	Double	Mvar	Reactive Power
S	Double	MVA	Apparent Power
cosphi	Double	1	Power Factor
u	Double	%	Generator Voltage Percentage
Zero- and Negative-Phase Sequence			
Flag_Z0	Integer		Grounding 0: Not grounded 1: Fixed grounded 2: Grounded w. impedances
Flag_Z0Input	Integer		Zero-Phase Sequence Input Data 1: Z0/Z1 and R0/X0 2: R0 and X0
R0_X0	Double	pu	Ratio R/X – Zero-Phase Sequence
Z0_Z1	Double	pu	Ratio Zero-Phase to Positive-Phase Sequence Impedance
R0	Double	Ohm	Resistance – Zero-Phase Sequence

X0	Double	Ohm	Reactance – Zero-Phase Sequence
X22	Double	%	Saturated Reactance – Negative-Phase Sequence
R2_X2	Double	pu	Ratio R/X – Negative-Phase Sequence
Controller Data			
Flag_Roh	Integer		State – Tap Position 1: Fixed 2: Variable
roh	Double	1	Present Tap Position
rohl	Double	1	Minimum Tap Position
rohu	Double	1	Maximum Tap Position
alpha	Double	°	Surplus Voltage Angle
Unode	Double	%	Controlled Voltage at Controller Node
Control Range Data			
Pmin	Double	MW	Active Power – Lower Limit
Pmax	Double	MW	Active Power – Upper Limit
Qmin	Double	Mvar	Reactive Power – Lower Limit
Qmax	Double	Mvar	Reactive Power – Upper Limit
Umin	Double	%	Voltage Lower Limit
Umax	Double	%	Voltage Upper Limit
cosphi_lim	Double		Limit Power Factor

Infeeder (Infeeder)

Attribute name	Data type	Unit	Description
Flag_Typ	Integer		State – Input Values 1: R and X 2: R/X and Sk2
R_X	Double	pu	Resistance/Reactance
xi	Double	%	Internal Reactance
Flag_LF	Integer		Load Flow Type 1: |I| and phi 2: P and Q 3: |usrc| and delta 4: |S| and cosphi 5: P and |u| 6: |Usrc| and delta 7: P and |U| 8: |uterm| and delta 9: |Uterm| and delta
I	Double	kA	Basic Current Source
P	Double	MW	Active Power
Q	Double	Mvar	Reactive Power
u	Double	%	Voltage
S	Double	MVA	Apparent Power
cosphi	Double	1	Power Factor
Ug	Double	kV	Generator Voltage
Zero- and Negative-Phase Sequence			
Flag_Z0	Integer		Grounding 0: Not grounded 1: Fixed grounded 2: Grounded w. impedances

Flag_Z0Input	Integer		Zero-Phase Sequence Input Data 1: Z0/Z1 and R0/X0 2: R0 and X0
Z0_Z1	Double	pu	Ratio Zero-Phase to Positive-Phase Sequence Impedance
R0_X0	Double	pu	Ratio R/X – Zero-Phase Sequence
R0	Double	Ohm	Resistance – Zero-Phase Sequence
X0	Double	Ohm	Reactance – Zero-Phase Sequence
Controller Data			
Unode	Double	%	Controlled Voltage at Controller Node
Control Range Data			
Pmin	Double	MW	Active Power – Lower Limit
Pmax	Double	MW	Active Power – Upper Limit
Qmin	Double	Mvar	Reactive Power – Lower Limit
Qmax	Double	Mvar	Reactive Power – Upper Limit
Umin	Double	%	Voltage Lower Limit
Umax	Double	%	Voltage Upper Limit
cosphi_lim	Double		Limit Power Factor

DC-Infeeder (DCInfeeder)

Attribute name	Data type	Unit	Description
Flag_Input_Type	Integer		DC Input 1: P and Q 2: P and cosphi 3: Inverter
P	Double	MW	Active Power
Q	Double	Mvar	Reactive Power
cosphi	Double	1	Power Factor
DC_power	Double	kW	Installed DC-Power
fDC_power	Double	1	Factor Installed DC-Power
fP	Double	1	Multiplication Factor – Active Power
fQ	Double	1	Multiplication Factor – Reactive Power
DC_losses	Double	%	Losses until Inverter
Eta_Inverter	Double	%	Efficiency – Inverter
Q_Inverter	Double	%	Reactive Power Demand – Inverter
Ctrl_power	Double	W	Controller Power
Tr_UrNet	Double	kV	Rated Voltage Netside – Transformer
Tr_Sr	Double	kVA	Rated Apparent Power – Transformer
Tr_uk	Double	%	Reference Short Circuit Voltage – Transformer
Tr_rx	Double	pu	Ratio R/X – Transformer
Umin_Inverter	Double	%	Minimum Voltage – Inverter
Umax_Inverter	Double	%	Maximum Voltage – Inverter
t_off	Double	s	Switch Off Time
Flag_Connect	Integer		Type of Connecting 1: Directly 2: Transformer
Flag_I	Flag_I		Control Current State 0: Not active at load flow 1: Active at load flow
Ireg	Ireg	kA	Control Current
pk	pk	1	Reference Compensation Power

Flag_PhiSC	Integer		Additional Short Circuit Data 0: No 1: Yes
PhiSC	Double	°	Angle Short Circuit Current
fSc	Double	1	Factor Short Circuit

Asynchronous Machine (AsynchronousMachine)

Attribute name	Data type	Unit	Description
Flag_Typ	Integer		Input Type of Asynchronous Machine 1: Pn 2: In 3: NEMA
Pn	Double	MW	Rated Active Power
Un	Double	kV	Rated Voltage
Speedn	Double	1/min	Rated Speed
pol	Double	1	Pole Pair Number
cosphin	Double	pu	Rated Power Factor
etan	Double	pu	Rated Efficiency
IaIn	Double	pu	Current Ratio At Start-Up
R_X	Double	pu	Ratio R/X – Positive-Phase Sequence
Inm	Double	kA	Rated Current
Flag_LF	Integer		Load Flow Type 1: P and Q 2: P and cosphi 3: P/Pn and cosphi 4: U, I und cosphi 5: DFIG (P, Q and Slip)
P	Double	MW	Active Power
Q	Double	Mvar	Reactive Power
cosphi	Double	1	Power Factor
ppn	Double	pu	Utilization
I	Double	kA	Basic Current Source
Slip	Double	%	Slip
Flag_SC	Integer		Short Circuit Behavior 1: Ik" + ip / I1c + Iint 2: ip / I1c 3: Ignore
Zero- and Negative-Phase Sequence			
Flag_Z0	Integer		Grounding 0: Not grounded 1: Fixed grounded 2: Grounded w. impedances
Flag_Z0Input	Integer		Zero-Phase Sequence Input Data 1: Z0/Z1 and R0/X0 2: R0 and X0
Z0_Z1	Double	pu	Ratio Zero-Phase to Positive-Phase Sequence Impedance
R0_X0	Double	pu	Ratio R/X – Zero-Phase Sequence
R0	Double	Ohm	Resistance – Zero-Phase Sequence
X0	Double	Ohm	Reactance – Zero-Phase Sequence
Ia2In	Double	pu	Current Ratio At Start-Up
R2_X2	Double	pu	Ratio R/X – Negative-Phase Sequence
Characteristics			

TA	Double	s	Starting Time Power Unit Data
GD2	Double	Mpm²	Momentum Power Unit Data
Motor Start-Up			
Flag_StartUpCtrl	Integer		Start Up Control 0: None 1: Current 2: Auto transformer 3: Current and auto transformer 4: Capacitor 5: Current and capacitor
ConStart	Integer		Start-Up Circuitry 1: Star 2: Delta 3: Star/delta
ConRun	Integer		Characteristic Data Circuitry 1: Star 2: Delta
Reliability Data			
CustCnt	Long Integer	1	Number of Supplied Customers

Load (Load)

Attribute name	Data type	Unit	Description
Flag_LoadTyp	Integer		Load Type 1: Load 2: Customer load
Flag_LF	Integer		Load Input 1: P, Q and (u) 2: P, Q and (U) 3: S, cosphi and u 4: S, cosphi and U 5: I, cosphi and u 6: I, cosphi and U 7: P and I 8: E, cosphi and t 9: Eap and Eaq 10: Pi and Qi 11: P, cosphi and u 12: P, cosphi and U 13: Pi, Qi and (u) – star 14: Pij, Qij and (u) – delta 15: P, Q and (u) – delta
P	Double	MW	Active Power
Q	Double	Mvar	Reactive Power
u	Double	%	Voltage
Ul	Double	kV	Voltage
S	Double	MVA	Apparent Power
cosphi	Double	pu	Power Factor
I	Double	kA	Current
P1	Double	MW	Active Power Phase 1
Q1	Double	Mvar	Reactive Power Phase 1
P2	Double	MW	Active Power Phase 2
Q2	Double	Mvar	Reactive Power Phase 2
P3	Double	MW	Active Power Phase 3
Q3	Double	Mvar	Reactive Power Phase 3
P12	Double	MW	Active Power Phase 12

Q12	Double	Mvar	Reactive Power Phase 12
P23	Double	MW	Active Power Phase 23
Q23	Double	Mvar	Reactive Power Phase 23
P31	Double	MW	Active Power Phase 31
Q31	Double	Mvar	Reactive Power Phase 31
Zero- and Negative-Phase Sequence			
Flag_Z0	Integer		Grounding 0: Not grounded 1: Fixed grounded 2: Grounded w. impedances
Flag_Z0Input	Integer		Zero-Phase Sequence Input Data 1: Z0/Z1 and R0/X0 2: R0 and X0
Z0_Z1	Double	pu	Ratio Zero-Phase to Positive-Phase Sequence Impedance
R0_X0	Double	pu	Ratio R/X – Zero-Phase Sequence
R0	Double	Ohm	Resistance – Zero-Phase Sequence
X0	Double	Ohm	Reactance – Zero-Phase Sequence
Pneg	Double	MW	Active Power – Negative-Phase Sequence
Qneg	Double	Mvar	Reactive Power – Negative-Phase Sequence
Dynamic Data			
ResFlux1	Double	pu	Residual Flux Phase L1
ResFlux2	Double	pu	Residual Flux Phase L2
ResFlux3	Double	pu	Residual Flux Phase L3

Shunt Impedance (ShuntImpedance)

Attribute name	Data type	Unit	Description
Un	Double	kV	Rated Voltage
R	Double	Ohm	Resistance
X	Double	Ohm	Reactance
Zero-Phase Sequence			
Flag_Z0	Integer		Grounding 0: Not grounded 1: Fixed grounded 2: Grounded w. impedances
Flag_Z0Input	Integer		Zero-Phase Sequence Input Data 1: Z0/Z1 and R0/X0 2: R0 and X0
R0_X0	Double	pu	Ratio R/X – Zero-Phase Sequence
Z0_Z1	Double	pu	Ratio Zero-Phase to Positive-Phase Sequence Impedance
R0	Double	Ohm	Resistance – Zero-Phase Sequence
X0	Double	Ohm	Reactance – Zero-Phase Sequence
Dynamic Data			
ResFlux1	Double	pu	Residual Flux Phase L1
ResFlux2	Double	pu	Residual Flux Phase L2
ResFlux3	Double	pu	Residual Flux Phase L3

Shunt Reactor (ShuntReactor)

Attribute name	Data type	Unit	Description
Sn	Double	MVA	Rated Apparent Power
Un	Double	kV	Rated Voltage
Vfe	Double	kW	Iron Losses
Vcu	Double	kW	Copper Losses
Zero-Phase Sequence			
Flag_Z0	Integer		Grounding 0: Not grounded 1: Fixed grounded 2: Grounded w. impedances
Flag_Z0Input	Integer		Zero-Phase Sequence Input Data 1: Z0/Z1 and R0/X0 2: R0 and X0
R0_X0	Double	pu	Ratio R/X – Zero-Phase Sequence
Z0_Z1	Double	pu	Ratio Zero-Phase to Positive-Phase Sequence Impedance
R0	Double	Ohm	Resistance – Zero-Phase Sequence
X0	Double	Ohm	Reactance – Zero-Phase Sequence
Controller Data			
Flag_Roh	Integer		Controller State 1: Fix 2: Variable – node 3: Variable – terminal
roh	Integer	1	Present Tap Position
rohl	Integer	1	Minimum Tap Position
rohu	Integer	1	Maximum Tap Position
deltaS	Double		Additional Reactive Power
uul	Double	%	Voltage Upper Limit
ull	Double	%	Voltage Lower Limit
Qmin	Double	Mvar	Minimum Total Reactive Power
Qmax	Double	Mvar	Maximum Total Reactive Power
CosPhiMin	Double	1	Cosinus Phi Minimum
CosPhiMax	Double	1	Cosinus Phi Maximum
Dynamic Data			
ResFlux1	Double	pu	Residual Flux Phase L1
ResFlux2	Double	pu	Residual Flux Phase L2
ResFlux3	Double	pu	Residual Flux Phase L3

Shunt Capacitor (ShuntCondensator)

Attribute name	Data type	Unit	Description
Sn	Double	MVA	Rated Apparent Power
Un	Double	kV	Rated Voltage
Vdi	Double	kW	Dielectric Losses
Zero-Phase Sequence			

Flag_Z0	Integer		Grounding 0: Not grounded 1: Fixed grounded 2: Grounded w. impedances
Flag_Z0Input	Integer		Zero-Phase Sequence Input Data 1: Z0/Z1 and R0/X0 2: R0 and X0
R0_X0	Double	pu	Ratio R/X – Zero-Phase Sequence
Z0_Z1	Double	pu	Ratio Zero-Phase to Positive-Phase Sequence Impedance
R0	Double	Ohm	Resistance – Zero-Phase Sequence
X0	Double	Ohm	Reactance – Zero-Phase Sequence
Controller Data			
Flag_Roh	Integer		Controller State 1: Fix 2: Variable – node 3: Variable – terminal
roh	Integer	1	Present Tap Position
rohl	Integer	1	Minimum Tap Position
rohu	Integer	1	Maximum Tap Position
deltaS	Double		Additional Reactive Power
uul	Double	%	Voltage Upper Limit
ull	Double	%	Voltage Lower Limit
Qmin	Double	Mvar	Minimum Total Reactive Power
Qmax	Double	Mvar	Maximum Total Reactive Power
CosPhiMin	Double	1	Cosinus Phi Minimum
CosPhiMax	Double	1	Cosinus Phi Maximum

Variable Shunt Element (VarShuntElement)

Attribute name	Data type	Unit	Description
Flag_LF	Integer		Load Flow Input 1: Power 2: Impedance 3: Model 4: Mixed power 5: Function
Flag_LoadType	Integer		Load Flow Type 1: Z constant 2: P and Q constant 3: I constant
Flag_Macro_LF	Integer		Model Type Load Flow 0: None 1: Controller 2: Equivalent circuit
Plf	Double	MW	Active Power Load Flow
Qlf	Double	Mvar	Reactive Power Load Flow
Ulf	Double	kV	Voltage Load Flow
Rlf	Double	Ohm	Resistance Load Flow
Xlf	Double	Ohm	Reactance Load Flow
fPk	Double	pu	Factor Constant Active Power
fPi	Double	pu	Factor Current Dependent Active Power
fPu	Double	pu	Factor Voltage Dependent Active Power
fQk	Double	pu	Factor Constant Reactive Power
fQi	Double	pu	Factor Current Dependent Reactive Power
fQu	Double	pu	Factor Voltage Dependent Reactive Power

f_p_1	Double	pu	Factor 1 Active Power
f_q_1	Double	pu	Factor 1 Reactive Power
e_p_1	Double	pu	Exponent 1 Active Power
e_q_1	Double	pu	Exponent 1 Reactive Power
f_p_2	Double	pu	Factor 2 Active Power
f_q_2	Double	pu	Factor 2 Reactive Power
e_p_2	Double	pu	Exponent 2 Active Power
e_q_2	Double	pu	Exponent 2 Reactive Power
f_p_3	Double	pu	Factor 3 Active Power
f_q_3	Double	pu	Factor 3 Reactive Power
e_p_3	Double	pu	Exponent 3 Active Power
e_q_3	Double	pu	Exponent 3 Reactive Power
Rsc	Double	Ohm	Resistance Circuit Input
Xsc	Double	Ohm	Reactance Circuit Input
Zero- and Negative-Phase Sequence			
Flag_Z0Input	Integer		Zero-Phase Sequence Input Data 1: Z0/Z1 and R0/X0 2: R0 and X0
Z0_Z1	Double	pu	Ratio Zero-Phase to Positive-Phase Sequence Impedance
R0_X0	Double	pu	Ratio R/X – Zero-Phase Sequence
R0	Double	Ohm	Resistance – Zero-Phase Sequence
X0	Double	Ohm	Reactance – Zero-Phase Sequence
Pneg	Double	MW	Active Power – Negative-Phase Sequence
Qneg	Double	Mvar	Reactive Power – Negative-Phase Sequence

Shunt Harmonics Resonance Network (HarResNet)

Attribute name	Data type	Unit	Description
Un	Double	kV	Rated Voltage
R	Double		Resistance at Network Frequency
X	Double		Reactance at Network Frequency
Factor	Double		Initial Value Factor
Impedance	Integer		Determine Impedance 1: Vmax 2: Imax
FlagZ0	Integer		Zero Sequence Data 0: Not grounded 1: Fixed grounded

Two-Winding Transformer (TwoWindingTransformer)

Attribute name	Data type	Unit	Description
Un1	Double	kV	Rated Voltage (Side 1)
Un2	Double	kV	Rated Voltage (Side 2)
Sn	Double	MVA	Rated Apparent Power
Smax	Double	MVA	Full Load Power
Smax1	Double	MVA	First Additional Full Load Power
Smax2	Double	MVA	Second Additional Full Load Power

Smax3	Double	MVA	Third Additional Full Load Power
uk	Double	%	Reference Short Circuit Voltage
ur	Double	%	Short Circuit Voltage – Ohmic Part
Vfe	Double	kW	Iron Losses
i0	Double	%	No Load Current
VecGrp	Integer		Vector Group 1: DD0, 2: DZ0, 3: DZN0, 4: YNY0, 5: YNYN0, 6: YY0, 7: YYN0, 8: ZD0, 9: ZND0, 10: DYN1, 11: DZ1, 12: DZN1, 13: YD1, 14: YND1, 15: YNZN1, 16: YZ1, 17: YZN1, 18: ZD1, 19: ZND1, 20: ZNYN1, 21: ZY1, 22: ZYN1, 23: DY5, 24: DYN5, 25: YD5, 26: YND5, 27: YNZ5, 28: YNZN5, 29: YZ5, 30: YZN5, 31: ZNY5, 32: ZNYN5, 33: ZY5, 34: ZYN5, 35: DD6, 36: DZ6, 37: DZN6, 38: YNY6, 39: YNYN6, 40: YY6, 41: YYN6, 42: ZD6, 43: ZND6, 44: DY7, 45: DYN7, 46: DZ7, 47: DZN7, 48: YD7, 49: YND7, 50: YNZN7, 51: YZ7, 52: YZN7, 53: ZD7, 54: ZND7, 55: ZNYN7, 56: ZY7, 57: ZYN7, 58: DY11, 59: DYN11, 60: YD11, 61: YND11, 62: YNZ11, 63: YNZN11, 64: YZ11, 65: YZN11, 66: ZNY11, 67: ZNYN11, 68: ZY11, 69: ZYN11, 70: DY1, 71: Y0, 72: YN0, 73: D0, 74: ZNY1, 75: ZNY7, 76: DDN0, 77: DND0, 78: DNYN1, 79: DNYN11, 80: YNDN1, 81: YNDN11
uk_ct	Double	%	Ref. Short Circuit Voltage Half Winding
ur_ct	Double	%	SC Voltage – Ohmic Part Half Winding
Zero-Phase Sequence			
FlagZ0Input	Integer		Zero Data Input 1: Z0/Z1 and R0/X0 2: R0 and X0 3: R0/R1 and X0/X1 4: ZABNL, ZBANL and ZABSC
Z0_Z1	Double	pu	Ratio Zero-Phase to Positive-Phase Sequence Impedance
R0_R1	Double	pu	Ratio Zero-Phase to Positive-Phase Resistance
R0	Double	Ohm	Resistance – Zero-Phase Sequence
X0	Double	Ohm	Reactance – Zero-Phase Sequence
X0_X1	Double	pu	Ratio Zero-Phase to Positive-Phase Reactance
R0_X0	Double	pu	Ratio R/X – Zero-Phase Sequence
ZABNL	Double	Ohm	Impedance between A and B in No Load
ZBANL	Double	Ohm	Impedance between B and A in No Load
ZABSC	Double	Ohm	Impedance between B and A in Short Circuit
Controller Data			
FlagConNode	Integer		Controller Node 1: Side 1 2: Side
Flag_Roh	Integer		State – Tap Position 1: Fixed 2: Node 3: Impedance 4: Active power 5: Reactive power 6: Control Charact.
Flag_Tap	Integer		Individual Tap Positions 0: No 1: Yes
roh	Double		Present Tap Position
roh1	Double		Present Tap Position Winding 1
roh2	Double		Present Tap Position Winding 2
roh3	Double		Present Tap Position Winding 3
rohl	Double		Minimum Tap Position
rohm	Double		Main Tap Position

rohu	Double		Maximum Tap Position
alpha	Double	°	Additional Voltage Angle
ukr	Double	%	Additional Voltage per Tap Position
phi	Double	°	Voltage Phase Shift per Tap Position
ukl	Double	%	Short Circuit Voltage at Minimum Tap Position
uku	Double	%	Short Circuit Voltage at Maximum Tap Position
ull	Double	%	Voltage Lower Limit
uul	Double	%	Voltage Upper Limit
Plp	Double	MW	Active Power Lower Limit for Controller
Pup	Double	MW	Active Power Upper Limit for Controller
Qlp	Double	Mvar	Reactive Power Lower Limit for Controller
Qup	Double	Mvar	Reactive Power Upper Limit for Controller
Dynamic Data			
ResFlux1	Double	pu	Residual Flux Phase L1
ResFlux2	Double	pu	Residual Flux Phase L2
ResFlux3	Double	pu	Residual Flux Phase L3

Three-Winding Transformer (ThreeWindingTransformer)

Attribute name	Data type	Unit	Description
Un1	Double	kV	Rated Voltage (Side 1)
Un2	Double	kV	Rated Voltage (Side 2)
Un3	Double	kV	Rated Voltage (Side 3)
Sn12	Double	MVA	Rated Apparent Through Power (Side 1-2)
Sn23	Double	MVA	Rated Apparent Through Power (Side 2-3)
Sn31	Double	MVA	Rated Apparent Through Power (Side 3-1)
Smax1	Double	MVA	Full Load Power (Side 1)
Smax2	Double	MVA	Full Load Power (Side 2)
Smax3	Double	MVA	Full Load Power (Side 3)
Smax1_1	Double	MVA	First Additional Full Load Power (Side 1)
Smax1_2	Double	MVA	Second Additional Full Load Power (Side 1)
Smax1_3	Double	MVA	Third Additional Full Load Power (Side 1)
Smax2_1	Double	MVA	First Additional Full Load Power (Side 2)
Smax2_2	Double	MVA	Second Additional Full Load Power (Side 2)
Smax2_3	Double	MVA	Third Additional Full Load Power (Side 2)
Smax3_1	Double	MVA	First Additional Full Load Power (Side 3)
Smax3_2	Double	MVA	Second Additional Full Load Power (Side 3)
Smax3_3	Double	MVA	Third Additional Full Load Power (Side 3)
uk1	Double	%	Reference Short Circuit Voltage (Side 1-2)
uk2	Double	%	Reference Short Circuit Voltage (Side 2-3)
uk3	Double	%	Reference Short Circuit Voltage (Side 3-1)
ur1	Double	%	Ohmic Short Circuit Voltage (Side 1-2)
ur2	Double	%	Ohmic Short Circuit Voltage (Side 2-3)
ur3	Double	%	Ohmic Short Circuit Voltage (Side 3-1)
i0	Double	%	No Load Current
Vfe	Double	kW	Iron Losses
phi1	Double	°	Additional Phase Rotation (Side 1)
phi2	Double	°	Additional Phase Rotation (Side 2)
phi3	Double	°	Additional Phase Rotation (Side 3)

VecGrp1	Integer		Vector Group (Side 1) 1: Y0, 2: YN0, 3: Y6, 4: YN6, 5: D1, 6: D5, 7: D7, 8: D11, 9: Z1, 10: ZN1, 11: Z5, 12: ZN5, 13: Z7, 14: ZN7, 15: Z11, 16: ZN11, 17: ATN, 18: AT
VecGrp2	Integer		Vector Group (Side 2) 1: Y0, 2: YN0, 3: Y6, 4: YN6, 5: D1, 6: D5, 7: D7, 8: D11, 9: Z1, 10: ZN1, 11: Z5, 12: ZN5, 13: Z7, 14: ZN7, 15: Z11, 16: ZN11, 17: ATN, 18: AT
VecGrp3	Integer		Vector Group (Side 3) 1: Y0, 2: YN0, 3: Y6, 4: YN6, 5: D1, 6: D5, 7: D7, 8: D11, 9: Z1, 10: ZN1, 11: Z5, 12: ZN5, 13: Z7, 14: ZN7, 15: Z11, 16: ZN11
Zero-Phase Sequence			
FlagZ0Input	Integer		Input 1: Z0/Z1 and R0/X0 2: R0 and X0 3: R0/R1 and X0/X1
Z0_Z1_12	Double	pu	Ratio Zero-Phase to Positive-Phase Sequence Impedance
Z0_Z1_23	Double	pu	Ratio Zero-Phase to Positive-Phase Sequence Impedance
Z0_Z1_31	Double	pu	Ratio Zero-Phase to Positive-Phase Sequence Impedance
R0_R1_12	Double	pu	Ratio Zero-Phase to Positive-Phase Resistance
R0_R1_23	Double	pu	Ratio Zero-Phase to Positive-Phase Resistance
R0_R1_31	Double	pu	Ratio Zero-Phase to Positive-Phase Resistance
R0_12	Double	Ohm	Resistance – Zero-Phase Sequence
R0_23	Double	Ohm	Resistance – Zero-Phase Sequence
R0_31	Double	Ohm	Resistance – Zero-Phase Sequence
X0_12	Double	Ohm	Reactance – Zero-Phase Sequence
X0_23	Double	Ohm	Reactance – Zero-Phase Sequence
X0_31	Double	Ohm	Reactance – Zero-Phase Sequence
X0_X1_12	Double	pu	Ratio Zero-Phase to Positive-Phase Reactance
X0_X1_23	Double	pu	Ratio Zero-Phase to Positive-Phase Reactance
X0_X1_31	Double	pu	Ratio Zero-Phase to Positive-Phase Reactance
R0_X0_12	Double	pu	Ratio R/X – Zero-Phase Sequence
R0_X0_23	Double	pu	Ratio R/X – Zero-Phase Sequence
R0_X0_31	Double	pu	Ratio R/X – Zero-Phase Sequence
Controller Data			
Flag_Roh1	Integer		State – Tap Position (Side 1) 0: None 1: Fixed 2: Node 3: Impedance 4: Active power 5: Reactive power
Flag_Roh2	Integer		State – Tap Position (Side 2) 0: None 1: Fixed 2: Node 3: Impedance 4: Active power 5: Reactive power
Flag_Roh3	Integer		State – Tap Position (Side 3) 0: None 1: Fixed 2: Node 3: Impedance 4: Active power 5: Reactive power
roh1	Double		Present Tap Position (Side 1)

roh2	Double		Present Tap Position (Side 2)
roh3	Double		Present Tap Position (Side 3)
rohu1	Double		Maximum Tap Position (Side 1)
rohu2	Double		Maximum Tap Position (Side 2)
rohu3	Double		Maximum Tap Position (Side 3)
rohm1	Double		Main Tap Position (Side 1)
rohm2	Double		Main Tap Position (Side 2)
rohm3	Double		Main Tap Position (Side 3)
rohl1	Double		Minimum Tap Position (Side 1)
rohl2	Double		Minimum Tap Position (Side 2)
rohl3	Double		Minimum Tap Position (Side 3)
Dynamic Data			
ResFlux1	Double	pu	Residual Flux Phase L1
ResFlux2	Double	pu	Residual Flux Phase L2
ResFlux3	Double	pu	Residual Flux Phase L3

Line (Line)

Attribute name	Data type	Unit	Description
Flag_LineTyp	Integer		Line Type 1: Cable 2: Overhead line 3: Connector
FlagMat	Integer		Line Material 1: Al 2: Cu
Len	Double	km	Length
ParSys	Double	1	Number of Parallel Systems
R	Double	Ohm/km	Resistance
X	Double	Ohm/km	Reactance
C	Double	nF/km	Capacitance
va	Double	kW/km	Leakage Losses Ground Fault
Ith	Double	kA	Thermal Limit Current
Ith1	Double	kA	First Additional Limit Current
Ith2	Double	kA	Second Additional Limit Current
Ith3	Double	kA	Third Additional Limit Current
FrqNenn	Double	Hz	Rated Frequency
alpha	Double	1/°C	Temperature Coefficient for Temperature Dependent Resistance Change
Zero-Phase Sequence			
Flag_Z0Input	Integer		Zero-Phase Sequence Input Data 1: X0/X1 and R0/R1 2: r0 and x0
R0_R1	Double	pu	Ratio Zero-Phase to Positive-Phase Resistance
X0_X1	Double	pu	Ratio Zero-Phase to Positive-Phase Reactance
R0	Double	Ohm/km	Resistance – Zero-Phase Sequence
X0	Double	Ohm/km	Reactance – Zero-Phase Sequence
C0	Double	nF/km	Capacitance in Zero-Phase Sequence
rR	Double	Ohm/km	Resistance – Return Conductor
xR	Double	Ohm/km	Reactance – Return Conductor

Variable Serial Element (VarSerialElement)

Attribute name	Data type	Unit	Description
Flag_LF	Integer		Load Flow Input 1: Impedance 2: Model
Flag_Macro_LF	Integer		Model Type Load Flow 0: None 1: Controller 2: Equivalent circuit
Ur1	Double	kV	Rated Voltage Side 1
Ur2	Double	kV	Rated Voltage Side 2
R12lf	Double	Ohm	Resistance Load Flow
X12lf	Double	Ohm	Reactance Load Flow
R21lf	Double	Ohm	Resistance Load Flow
X21lf	Double	Ohm	Reactance Load Flow
R12sc	Double	Ohm	Resistance Short Circuit
X12sc	Double	Ohm	Reactance Short Circuit
R21sc	Double	Ohm	Resistance Short Circuit
X21sc	Double	Ohm	Reactance Short Circuit
Zero-Phase Sequence			
Flag_Z0Input	Integer		Zero-Phase Sequence Input Data 1: Z0/Z1 and R0/X0 2: R0 and X0
R0_X0	Double	pu	Ratio R/X – Zero-Phase Sequence
Z0_Z1	Double	pu	Ratio Zero-Phase to Positive-Phase Sequence Impedance
R0	Double	Ohm	Resistance – Zero-Phase Sequence
X0	Double	Ohm	Reactance – Zero-Phase Sequence
Dynamic Data			
Flag_Macro_SC	Integer		Impedances for Dynamics 1: Load flow 2: Short circuit
Harmonics			
Flag_Har	Integer		State – Harmonics 0: No frequency dependency 1: Quality – R constant 2: Quality – X/R constant 3: Impedance characteristic
qr	Double	1	Quality – R Constant
ql	Double	1	Quality – X/R Constant

Serial Reactor (SerialReactor)

Attribute name	Data type	Unit	Description
Flag_CoInput	Integer		Input Data 1: Reference coil voltage 2: Inductance
uD	Double	%	Reference Coil Voltage
L	Double	mH	Inductance
Un	Double	kV	Rated Voltage
InD	Double	kA	Rated Current

Ith1	Double	kA	First Additional Limit Current
Ith2	Double	kA	Second Additional Limit Current
Ith3	Double	kA	Third Additional Limit Current
Zero-Phase Sequence			
R_X	Double		Ratio R/X – Positive-Phase Sequence
Flag_Z0Input	Integer		Zero-Phase Sequence Input Data 1: R0/R1 and X0/X1 2: R0 and X0 3: R0 and L0
X0_X1	Double		Ratio Zero-Phase to Positive-Phase Reactance
R0_R1	Double		Ratio Zero-Phase to Positive-Phase Resistance
R0	Double	Ohm	Resistance – Zero-Phase Sequence
X0	Double	Ohm	Reactance – Zero-Phase Sequence
L0	Double	mH	Inductance in Zero-Phase Sequence
Dynamic Data			
ResFlux1	Double	pu	Residual Flux Phase L1
ResFlux2	Double	pu	Residual Flux Phase L2
ResFlux3	Double	pu	Residual Flux Phase L3

Serial Capacitor (SerialCondensator)

Attribute name	Data type	Unit	Description
C	Double	nF/km	Capacitance
XC	Double	Ohm	Capacitive Reactance
Un	Double	kV	Rated Voltage
R_X	Double	pu	Ratio R/X – Positive-Phase Sequence
Smax	Double	MVA	Full Load Power
Smax1	Double	MVA	First Additional Full Load Power
Smax2	Double	MVA	Second Additional Full Load Power
Smax3	Double	MVA	Third Additional Full Load Power
Zero-Phase Sequence			
Flag_Z0Input	Integer		Zero-Phase Sequence Input Data 1: R0/R1 and X0/X1 2: R0 and X0 3: R0 and C0
X0_X1	Double		Ratio Zero-Phase to Positive-Phase Reactance
R0_R1	Double		Ratio Zero-Phase to Positive-Phase Resistance
R0	Double	Ohm	Resistance – Zero-Phase Sequence
X0	Double	Ohm	Reactance – Zero-Phase Sequence

Serial Harmonics Resonance Network (HarBranchResNet)

Attribute name	Data type	Unit	Description
Un	Double	kV	Rated Voltage
R1	Double	Ohm	Resistance at Network Frequency
X1	Double	Ohm	Reactance at Network Frequency

Impedance	Integer		Determine Impedance 1: Vmax 2: Imax
RCData	Integer		Ripple Control Impedance 0: No 1: Yes
R1rc	Double	Ohm	Resistance at Ripple Control Frequency
X1rc	Double	Ohm	Reactance at Ripple Control Frequency
FlagZ0	Integer		Input Data Zero-Phase Sequence System 1: Blocking 2: Z0 identical Z1 3: R0/R1 and X0/X1 4: R0 and X0
R0	Double	Ohm	Resistance Zero-Phase Sequence System
X0	Double	Ohm	Reactance Zero-Phase Sequence System
R0_R1	Double	1	Ratio Zero-Phase to Positive-Phase Resistance
X0_X1	Double	1	Ratio Zero-Phase to Positive-Phase Reactance

Fault Observation (ProtOCFault)

Attribute name	Data type	Unit	Description
Node_ID	Long Integer		Sets the node
Element_ID	Long Integer		Sets the branch
Flag_State	Integer		Operating State 0: Off 1: On
Flag_FaultPhase	Integer		Faulty Phases 1: L1 2: L2 3: L3 4: L23 5: L31 6: L12 7: L123 0: None
Flag_InterruptPhase	Integer		Interrupted Phases 1: L1 2: L2 3: L3 6: L12 4: L23 5: L31 7: L123 0: None 8: N 9: L123N
len	Double		Distance
Flag_FaultReturn	Integer		Fault to Return Conductor 1: Short Circuit 2: Return Circuit 3: Ground Circuit 4: Return and Ground Circuit
Flag_FaultGround	Integer		Fault to Ground 1: Short Circuit 2: Return Circuit 3: Ground Circuit 4: Return and Ground Circuit

Flag_RefPhase	Integer		Reference Phase 0: None 1: L1 2: L2 3: L3
Flag_CondFaultOn	Integer		Conditions Fault On 0: None 1: Default 2: Time 3: Voltage 4: Voltage and time delay
ton	Double		Time On
On_NodeID	Long Integer		On Node
Flag_PhaseOn	Integer		On Phase 1: L1 2: L2 3: L3
Flag_Val	Integer		On Value 1: Minimum 2: Maximum 3: User-defined
Uon	Double		On Voltage
dT1	Double		On Time Delay – Next Phase
dT2	Double		On Time Delay – Previous Phase
Flag_CondFaultOff	Integer		Conditions Fault Off 0: None 1: Default 2: Time 3: Current 4: Current and time delay
toff	Double		Time Off
Current	Double		Off Current

Protection Location (ProtLocation)

Attribute name	Data type	Unit	Description
Flag_State	Integer		Active 0: Off 1: On

4.2.4 Attributes of Calculation Objects for Pipe Networks

Water Networks

Calculation Settings (FlowCalcParameter)

Attribute name	Data type	Unit	Description
ITmax	Integer		Maximum Number of Iterations (non-linear)
ITmax2	Integer		Maximum Number of Iterations (linear)
MeshAccuracy	Double	bar	Mesh Accuracy
NodeAccuracy	Double	l/s	Node Accuracy
FlowStep	Double	l/s	Maximum Step for Flow
Flag_Operate	Integer		Check Operating Conditions 0: Warning 1: Error
fCharCurve	Double	pu	Characteristic Curve Factor
SpecDensity	Double	kg/m³	Specific Density
KinematicVis	Double	mm²/s	Kinematic Viscosity
Flag_Pump	Integer		Parallel Pumps 0: No 1: Yes
Flag_Result	Integer		Store Results in Database 0: None 1: All 2: Restricted elements only 3: All elements in case of restrictions
StartTime	Double	s	Starting Time
Duration	Double	s	Duration
TimeStepGeo	Double	s	Time Step Geo-stationary

Network Level (FlowVoltageLevel)

Attribute name	Data type	Unit	Description
pRated	Double	bar	Rated Pressure
vMax	Double	m/s	Maximum Flow Velocity
pMin	Double	bar	Minimum Operating Pressure
pMax	Double	bar	Maximum Operating Pressure

Network Area (FlowNetworkGroup)

Attribute name	Data type	Unit	Description
Flag_MarkedForCaused	Integer		Marked for Caused Malfunction 0: No 1: Yes
Flag_Malfunc	Integer		Malfunction 0: None 1: All elements 2: All lines 3: All restricted elements 4: All restricted lines
Speed_BaseLimit	Double	m/s	Base Speed Limit

Flag_CausedMalfunc	Integer		Caused Malfunction 0: None 1: Marked areas 2: Own area
Flag_CausedElem	Integer		Caused Elements 1: Restricted elements 2: Restricted lines
Speed_CausedLimit	Double	m/s	Caused Speed Limit
Flag_Report	Integer		Reporting 0: None 1: Elements and nodes 2: Lines and nodes 3: Elements 4: Lines 5: Nodes
Flag_FireWater	Integer		Join Fire Water Simulation 0: No 1: Yes
ConLineLength	Double	m	Length
ConLineDiameter	Double	mm	Diameter
ConLineRoughness	Double	mm	Sand Roughness
ConLineZeta	Double		Loss Factor Zeta Value
dsh	Double	m	Delta Elevation
QFireWater	Double	l/s	Fire Water Flow
pFireWater	Double	bar	Fire Water Pressure
tFireWater	Double	h	Fire Water Time
pRelMinLimit	Double	bar	Minimum Pressure – Relative

Const. Pressure Decrease/Const. Flow (FlowConstLine)

Attribute name	Data type	Unit	Description
Flag_Typ	Integer		Line Type 1: Constant pressure drop 2: Constant flow
PressureDecr	Double	bar	Pressure Drop

Consumer (FlowConsumer)

Attribute name	Data type	Unit	Description
Q	Double	l/s	Const. Consumption
Flag_ConControl	Integer		Pressure Dependent Consumption Decrease 0: No 1: Yes
pDiffMin	Double	bar	Minimum Pressure Difference
pRelMin	Double	bar	Minimum Relative Pressure
DesignTemp	Double	°C	Design Temperature of Consumer

Pressure Regulator (FlowPressureReg)

Attribute name	Data type	Unit	Description
pInlet	Double	bar	Max. Pressure Deviation
pOutlet	Double	bar	Pressure at Outlet Node

pDevation	Double	bar	Pressure at Inlet Node
Flag_PessInc	Integer		Function 1: Pressure increase 2: Pressure drop 3: Pressure increase and drop

Pressure Increase Pump (FlowPumpLine)

Attribute name	Data type	Unit	Description
Flag_Type	Integer		Pump Type 1: Centrifugal pump 2: Reciprocating pump
QOutput	Double	l/s	Output Flow
uPump	Double	1/min	Characteristic Pump Speed
FlowStep	Double	l/s	Maximum Flow

Sliding Valve/Non-Return Valve (FlowValve)

Attribute name	Data type	Unit	Description
Flag_Type	Integer		Valve Type 1: Sliding valve 2: Non-return valve
Opening	Double	%	Degree of Opening
Diameter	Double	mm	Valve Diameter
Pos	Integer		Valve Position 0: Close 1: Open

Leakage (FlowLeakage)

Attribute name	Data type	Unit	Description
OutputSurface	Double	mm²	Output Surface
fFlow	Double	pu	Flow Number
FlowStep	Double	l/s	Maximum Step for Flow
ConLineLength	Double	m	Connection Line Length
ConLineDiameter	Double	mm	Connection Line Diameter
ConLineRoughness	Double	mm	Connection Line Sand Roughness
ConLineZeta	Double		Connection Line Sand Zeta Value
dsh	Double	m	Delta Elevation
QFireWater	Double	l/s	Fire Water Flow
pFireWater	Double	bar	Fire Water Pressure
tFireWater	Double	h	Fire Water Time

Pressure Buffer (FlowPressureBuffer)

Attribute name	Data type	Unit	Description
PMax	Double	bar	Maximum Pressure

Pump (FlowPump)

Attribute name	Data type	Unit	Description
Flag_Type	Integer		Pump Type 1: Centrifugal pump 2: Reciprocating pump
QOutput	Double	l/s	Output Flow
uPump	Double	1/min	Characteristic Pump Speed
FlowStep	Double	l/s	Maximum Flow
Flag_Limits	Integer		Limit Type 0: None 1: Flow
QOutputmin	Double	l/s	Minimum Output Flow
QOutputmax	Double	l/s	Maximum Output Flow

Water Tower (FlowWaterTower)

Attribute name	Data type	Unit	Description
hWaterLevel	Double	m	Water Level
hFillStart1	Double	m	Filling Level 1 Start
hFillStop1	Double	m	Filling Level 1 Stop
uPump1	Double	1/min	Pump Characteristics 1
hFillStart2	Double	m	Filling Level 1 Start
hFillStop2	Double	m	Filling Level 1 Stop
uPump2	Double	1/min	Pump Characteristics 1
hFillStart3	Double	m	Filling Level 1 Start
hFillStop3	Double	m	Filling Level 1 Stop
uPump3	Double	1/min	Pump Characteristics 1
Flag_Level	Integer		Level Data 0: No 1: Yes
Flag_Limits	Integer		Limit Type 0: None 1: Flow
Qmin	Double	l/s	Minimum Flow
Qmax	Double	l/s	Maximum Flow

Line (FlowLine)

Attribute name	Data type	Unit	Description
LineLength	Double	m	Length
Diameter	Double	mm	Diameter
SandRoughness	Double	mm	Sand Roughness
fLength	Double	%	Length Allowance Factor
fCurve	Double		Curve Factor
fDiameterAn	Double	%	Annual Diameter Reduction
fRoughnessAn	Double	%	Annual Roughness Increase
Zeta	Double		Zeta Value
LeakageRate	Double	l/sm	Leakage Rate

Node (FlowNode)

Attribute name	Data type	Unit	Description
Sh	Double	m	Elevation

Gas Networks

Calculation Settings (FlowCalcParameter)

Attribute name	Data type	Unit	Description
ITmax	Integer		Maximum Number of Iterations (non-linear)
ITmax2	Integer		Maximum Number of Iterations (linear)
MeshAccuracy	Double	bar	Mesh Accuracy
NodeAccuracyG	Double	m³/h	Node Accuracy
FlowStepG	Double	m³/h	Maximum Step for Flow
Flag_Operate	Integer		Check Operating Conditions 0: Warning 1: Error
SpecDensity	Double	kg/m³	Specific Density
HeatingAmount	Double	MJ/kg	Heating Amount
pAir	Double	bar	Air Pressure
SutherlandConst	Double	K	Sutherland Constant
AdiabaticExp	Double		Adiabatic Exponent
fConst	Double		Constant Factor
fLinear	Double		Linear Factor
Flag_Result	Integer		Store Results in Database 0: None 1: All 2: Restricted elements only 3: All elements in case of restrictions
StartTime	Double	s	Starting Time
Duration	Double	s	Duration
TimeStepGeo	Double	s	Time Step Geo-stationary

Network Level (FlowVoltageLevel)

Attribute name	Data type	Unit	Description
pRated	Double	bar	Rated Pressure
TGas	Double	°C	Gas Temperature
TAir	Double	°C	Air Temperature
vMax	Double	m/s	Maximum Flow Velocity
pMin	Double	bar	Minimum Operating Pressure
pMax	Double	bar	Maximum Operating Pressure

Network Area (FlowNetworkGroup)

Attribute name	Data type	Unit	Description
Flag_MarkedForCaused	Integer		Marked for Caused Malfunction 0: No 1: Yes
Flag_Malfunc	Integer		Malfunction 0: None 1: All elements 2: All lines 3: All restricted elements 4: All restricted lines
Speed_BaseLimit	Double	m/s	Base Speed Limit
Flag_CausedMalfunc	Integer		Caused Malfunction 0: None 1: Marked areas 2: Own area
Flag_CausedElem	Integer		Caused Elements 1: Restricted elements 2: Restricted lines
Speed_CausedLimit	Double	m/s	Caused Speed Limit
Flag_Report	Integer		Reporting 0: None 1: Elements and nodes 2: Lines and nodes 3: Elements 4: Lines 5: Nodes

Compressor (FlowCompressor)

Attribute name	Data type	Unit	Description
pInlet	Double	bar	Max. Pressure Deviation
pOutlet	Double	bar	Pressure at Outlet Node
pDevation	Double	bar	Pressure at Inlet Node

Const. Pressure Decrease/Const. Flow (FlowConstLine)

Attribute name	Data type	Unit	Description
Flag_Typ	Integer		Line Type 1: Constant pressure drop 2: Constant flow
PressureDecr	Double	bar	Pressure Drop
FlowGas	Double	mN^3/h	Flow

Consumer (FlowConsumer)

Attribute name	Data type	Unit	Description
Flag_Q	Integer		Consumption Type 1: Standard 2: Operating Conditions 3: Power
Q1	Double	m^3/h	Constant Consumption – Standard

Q2	Double	m³/h	Constant Consumption – Operating Cond.
Q3	Double	MW	Constant Consumption – Power
pDiffMin	Double	bar	Minimum Pressure Difference
pRelMin	Double	bar	Minimum Relative Pressure
DesignTemp	Double	°C	Design Temperature of Consumer

Pressure Regulator (FlowPressureReg)

Attribute name	Data type	Unit	Description
pInlet	Double	bar	Max. Pressure Deviation
pOutlet	Double	bar	Pressure at Outlet Node
pDevation	Double	bar	Pressure at Inlet Node
Flag_PessInc	Integer		Function 1: Pressure increase 2: Pressure drop 3: Pressure increase and drop
QReturn	Double	m³/h	Maximum Return Flow

Sliding Valve/Non-Return Valve (FlowValve)

Attribute name	Data type	Unit	Description
Flag_Type	Integer		Valve Type 1: Sliding valve 2: Non-return valve
Pos	Integer		Valve Position 0: Close 1: Open

Leakage (FlowLeakage)

Attribute name	Data type	Unit	Description
OutputSurface	Double	mm²	Output Surface
fFlow	Double	pu	Flow Number
FlowStpG	Double	m³/h	Maximum Step for Flow

Infeeder Gas (FlowInfeederG)

Attribute name	Data type	Unit	Description
Flag_Typ	Integer		Infeeder Type 1: Pressure supply 2: Flow supply
QReturn	Double	m³/h	Maximum Return Flow
pConst	Double	bar	Constant Excess Pressure
FlagQ	Integer		Flow Supply Type 1: Flow supply 2: Operating Conditions 3: Power
Q1	Double	m³/h	Constant Supply – Standard

Q2	Double	m³/h	Constant Supply – Operating Condition
Q3	Double	MW	Constant Supply – Power
Flag_Limits	Integer		Limit Type 0: None 1: Flow
Qmin	Double		Minimum Supply
Qmax	Double		Maximum Supply

Pressure Buffer (FlowPressureBuffer)

Attribute name	Data type	Unit	Description
PMax	Double	bar	Maximum Pressure

Line (FlowLine)

Attribute name	Data type	Unit	Description
LineLength	Double	m	Length
Diameter	Double	mm	Diameter
SandRoughness	Double	mm	Sand Roughness
fLength	Double	%	Length Allowance Factor
fCurve	Double		Curve Factor
fDiameterAn	Double	%	Annual Diameter Reduction
fRoughnessAn	Double	%	Annual Roughness Increase
Zeta	Double		Zeta Value

Node (FlowNode)

Attribute name	Data type	Unit	Description
Sh	Double	m	Elevation
Pres	Double	bar	Pressure Reservation

Heating/Cooling Networks

Calculation Settings (FlowCalcParameter)

Attribute name	Data type	Unit	Description
ITmax	Integer		Maximum Number of Iterations (non-linear)
ITmax2	Integer		Maximum Number of Iterations (linear)
MeshAccuracy	Double	bar	Mesh Accuracy
NodeAccuracy	Double	l/s	Node Accuracy
FlowStep	Double	l/s	Maximum Step for Flow
Flag_Operate	Integer		Check Operating Conditions 0: Warning 1: Error
qSpec	Double	J/kgK	Specific Thermal Capacity

Flag_MalFunc	Integer		Circuit for Malfunction 1: Supply line 2: Return line 3: Supply and return line
Flag_Result	Integer		Store Results in Database 0: None 1: All 2: Restricted elements only 3: All elements in case of restrictions
StartTime	Double	s	Starting Time
Duration	Double	s	Duration
TimeStepGeo	Double	s	Time Step Geo-stationary

Network Level (FlowVoltageLevel)

Attribute name	Data type	Unit	Description
pRated	Double	bar	Rated Pressure
TRated	Double	°C	Rated Temperature
TAir	Double	°C	Air Temperature
vMax	Double	m/s	Maximum Flow Velocity
pMin	Double	bar	Minimum Operating Pressure Supply Line
pMax	Double	bar	Maximum Operating Pressure Supply Line
TSupplyLine	Double	°C	Temperature Supply Line
pMinR	Double	bar	Minimum Operating Pressure Return Line
pMaxR	Double	bar	Maximum Operating Pressure Return Line
TReturnLine	Double	°C	Temperature Return Line

Network Area (FlowNetworkGroup)

Attribute name	Data type	Unit	Description
Flag_MarkedForCaused	Integer		Marked for Caused Malfunction 0: No 1: Yes
Flag_Malfunc	Integer		Malfunction 0: None 1: All elements 2: All lines 3: All restricted elements 4: All restricted lines
Speed_BaseLimit	Double	m/s	Base Speed Limit
Flag_CausedMalfunc	Integer		Caused Malfunction 0: None 1: Marked areas 2: Own area
Flag_CausedElem	Integer		Caused Elements 1: Restricted elements 2: Restricted lines
Speed_CausedLimit	Double	m/s	Caused Speed Limit
Flag_Report	Integer		Reporting 0: None 1: Elements and nodes 2: Lines and nodes 3: Elements 4: Lines 5: Nodes

Const. Pressure Decrease/Const. Flow (FlowConstLine)

Attribute name	Data type	Unit	Description
Flag_Typ	Integer		Line Type 1: Constant pressure drop 2: Constant flow
PressureDecr	Double	bar	Pressure Drop
FlowHeating	Double	t/h	Flow

Consumer (FlowConsumer)

Attribute name	Data type	Unit	Description
Q	Double	l/s	Const. Consumption
Flag_ConTyp	Integer		Consumption Type 1: Constant consumption 2: Constant power consumption 3: Sum of consumption and power
Q3	Double	MW	Constant Consumption – Power
Q4	Double	t/h	Constant Consumption
Power	Double	MW	Constant Consumption – Power
Flag_ConControl	Integer		Pressure Dependent Consumption Decrease 0: No 1: Yes
pDiffMin	Double	bar	Minimum Pressure Difference
Flag_Temp	Integer		Temperature Type 1: Return temperature 2: Difference of temperature
T	Double	°C	Temperature
DesignTemp	Double	°C	Design Temperature of Consumer

Heat Exchanger (FlowHeatExchanger)

Attribute name	Data type	Unit	Description
Flag_Typ	Integer		Heat Exchanger Type 1: Hydraulic uncoupling 2: Power apply
Flag_ConControl	Integer		Primary Pressure Dependent Consumption Decrease 0: No 1: Yes
Flag_Temp	Integer		Temperature Type 1: Return temperature 2: Difference of temperature sup – ret 3: Difference of temperature sec – prim
Flag_Maint	Integer		Pressure Maintenance Type 1: Medium pressure, difference and parts 2: Supply pressure and difference 3: Return pressure and difference 4: Pump data and parts 5: Supply pressure and pump data 6: Return pressure and pump data
Flag_Master	Integer		Leading Supply 0: No 1: Yes
Efficiency	Double	%	Efficiency
pDiffMin	Double	bar	Primary Minimum Pressure Difference

tPrim	Double	°C	Primary Temperature
tFeed	Double	°C	Secondary Supply Temperature
uPump	Double	1/min	Characteristic Pump Speed
FlowStep	Double	l/s	Maximum Step for Flow
Power	Double	MW	Power
pMedium	Double	bar	Medium Pressure
pSupRet	Double	bar	Difference Pressure
pSupplyMaint	Double	bar	Supply Pressure
pReturnMaint	Double	bar	Return Pressure
SupplyPart	Double	%	Part – Supply Pressure
ReturnPart	Double	%	Part – Return Pressure
QOutput	Double	l/s	Output Flow

Pressure Regulator (FlowPressureReg)

Attribute name	Data type	Unit	Description
pInlet	Double	bar	Max. Pressure Deviation
pOutlet	Double	bar	Pressure at Outlet Node
pDevation	Double	bar	Pressure at Inlet Node QReturn
Flag_PessInc	Integer		Function 1: Pressure increase 2: Pressure drop 3: Pressure increase and drop
Flag_PressDif	Integer		Difference Pressure Regulator 0: No 1: Yes
pSupRet	Double	bar	Difference Pressure

Pressure Increase Pump (FlowPumpLine)

Attribute name	Data type	Unit	Description
Flag_Type	Integer		Pump Type 1: Centrifugal pump 2: Reciprocating pump
QOutput	Double	l/s	Output Flow
uPump	Double	1/min	Characteristic Pump Speed
FlowStep	Double	l/s	Maximum Flow

Sliding Valve/Non-Return Valve (FlowValve)

Attribute name	Data type	Unit	Description
Flag_Type	Integer		Valve Type 1: Sliding valve 2: Non-return valve
Opening	Double	%	Degree of Opening
Diameter	Double	mm	Valve Diameter
Pos	Integer		Valve Position 0: Close 1: Open

Leakage (FlowLeakage)

Attribute name	Data type	Unit	Description
OutputSurface	Double	mm²	Output Surface
fFlow	Double	pu	Flow Number
FlowStep	Double	l/s	Maximum Step for Flow

Temperature Regulator (FlowThermoReg)

Attribute name	Data type	Unit	Description
tMin	Double	°C	Minimum Temperature
tMax	Double	°C	Maximum Temperature
TempAccuracy	Double	°C	Temperature Accuracy
FlowStep	Double	t/h	Maximum Step for Flow

Infeeder Heating/Cooling (FlowInfeederH)

Attribute name	Data type	Unit	Description
Flag_Typ	Integer		Infeeder Type 1: Pressure supply 2: Power Supply 3: Pressure maintenance
Flag_SupTyp	Integer		Power Supply Type 1: Constant supply 2: Constant supply power
Flag_ConControl	Integer		Pressure Dependent Supply Decrease 0: No 1: Yes
Flag_T	Integer		Temperature Type 1: Supply temperature 2: Difference of temperature
Flag_Maint	Integer		Pressure Maintenance Type 1: Medium pressure, difference and parts 2: Supply pressure and difference 3: Return pressure and difference 4: Pump data and parts 5: Supply pressure and pump data 6: Return pressure and pump data
Flag_Master	Integer		Leading Supply 0: No 1: Yes
T	Double	°C	Temperature
pDiffMin	Double	bar	Minimum Pressure Difference
uPump	Double	1/min	Characteristic Pump Speed
FlowStep	Double	l/s	Maximum Step for Flow
pSupply	Double	bar	Pressure Supply
Q	Double	t/h	Constant Supply Volume
Power	Double	MW	Constant Power Supply
pMedium	Double	bar	Medium Pressure
pSupRet	Double	bar	Difference Pressure
QOutput	Double	l/s	Output Flow
pSupplyMain	Double	bar	Supply Pressure
pReturnMain	Double	bar	Return Pressure

SupplyPart	Double	%	Part – Supply Pressure
ReturnPart	Double	%	Part – Return Pressure
Flag_Limits	Integer		Limit Type 0: None 1: Flow 2: Power
Qmin	Double	t/h	Minimum Flow
Qmax	Double	t/h	Maximum Flow
Pmin	Double	MW	Minimum Power
Pmax	Double	MW	Maximum Power

Pressure Buffer (FlowPressureBuffer)

Attribute name	Data type	Unit	Description
PMax	Double	bar	Maximum Pressure

Pump (FlowPump)

Attribute name	Data type	Unit	Description
Flag_Type	Integer		Pump Type 1: Centrifugal pump 2: Reciprocating pump
QOutput	Double	l/s	Output Flow
uPump	Double	1/min	Characteristic Pump Speed
tSupply	Double	°C	Supply Temperature
FlowStep	Double	l/s	Maximum Flow
Flag_Limits	Integer		Limit Type 0: None 1: Flow
QOutputmin	Double	l/s	Minimum Output Flow
QOutputmax	Qmax	l/s	Maximum Output Flow

Line (FlowLine)

Attribute name	Data type	Unit	Description
LineLength	Double	m	Length
Diameter	Double	mm	Diameter
SandRoughness	Double	mm	Sand Roughness
fLength	Double	%	Length Allowance Factor
fCurve	Double		Curve Factor
Zeta	Double		Zeta Value
LeakageRate	Double	l/sm	Leakage Rate
HeatingCond	Double	W/Mk	Thermal Conductivity

Node (FlowNode)

Attribute name	Data type	Unit	Description
Sh	Double	m	Elevation
PDiffMin	Double	bar	Minimum Pressure Difference

5 Reference

5.1 Documentation

The complete PSS SINCAL documentation is available as online help. You can also find these documents in the "Doc\English\Sincal" directory on the installation DVD as PDF files.

User Interface

For a comprehensive description of all the functions of the PSS SINCAL user interface, see the **System Manual**.

New users should refer to the chapter on **Using an Example to Work on a Network**. This outlines, step by step, how to create, process, calculate and display the results for an electrical network.

Simulation Procedure

The **technical manuals for electrical networks** contain detailed descriptions of the various calculation methods for electrical networks and their input data. These manuals have two parts:

- Input data:
  Description of the input data for all simulation procedures.
- Technical manual for the respective simulation procedure:
  Comprehensive descriptions of the simulation procedure. The manuals are titled according to the specific procedure: Load Flow, Short Circuit, etc.

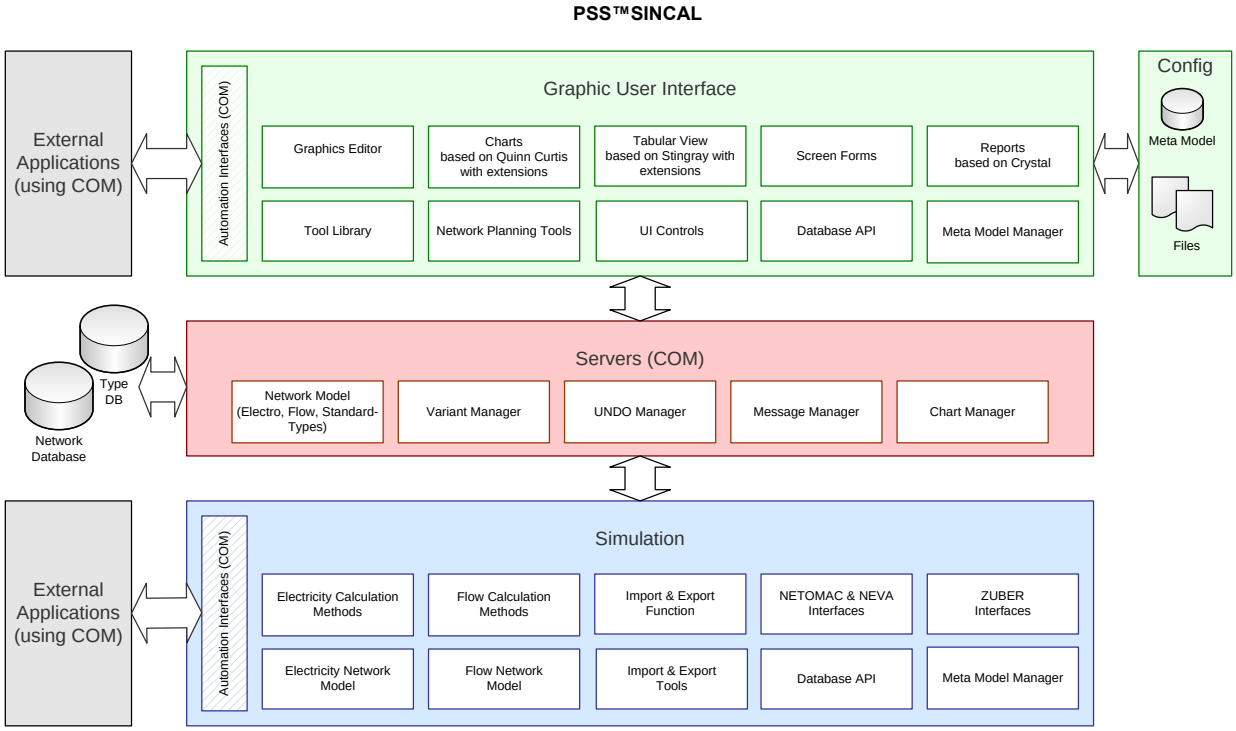
The **technical manuals for pipe networks** contain detailed descriptions of the various calculation methods for pipe networks and their input data. The following manuals are available: Water, Gas and Heating/Cooling.

Network Database

For detailed information about the network database structure, see the **Database Description Manual**. This manual presents the PSS SINCAL data model in detail. It explains both the structure and the semantics of the data model, and has detailed tables (relations) with their attributes.

5.2 PSS SINCAL Architecture

The following illustration shows the basic PSS SINCAL architecture.



As can be seen in the illustration above, there are two important components:

- **Graphic User Interface**
  This is the real PSS SINCAL user interface. This interface is used to process networks, modify data, evaluate the results, etc.
- **Simulation**
  This component contains all the PSS SINCAL calculation methods. The calculation component can also be used stand-alone (without the user interface).

Between these, the **Servers (COM)** component for internal data exchange is available. The entire database can be accessed with this component.

COM Interfaces

All PSS SINCAL components have COM interfaces. A differentiation is made between internal and external COM interfaces. The internal interfaces are only used within PSS SINCAL. They are not documented and as such not designed for external use. The external interfaces are documented and can also be used in your own applications. This lets you, for example, automize work sequences or integrate calculation methods into your own applications.

5.3 Ready-Made Solutions

In the area of GIS coupling, our partners offer ready-made solutions that can be tailored to meet your

specific needs. Varied PSS SINCAL clients have successfully implemented these solutions.

L&Mark Informatika Kft.

L&MARK Informatika Kft. is your partner in the implementation of PSS SINCAL GIS integration worldwide. The company is partner of AED-SICAD and has existing solutions on several GIS platform (ESRI ArcFM, ArcFM-UT, Sicad/open, SICAD-UTE, Intergraph GNET, etc.).

L&MARK Informatika Kft.
Mrs Andrea Lisziewicz
Margit krt. 43-45. 4/5.
1024 Budapest
Hungaria
Fon +36 1 201 7725
Fax +36 1 201 2817
e-mail andrea.lisziewicz@lmark.hu
http://www.lmark.hu

Mettenmeier GmbH

Mettenmeier GmbH is a software consultant, service and solution provider with corporate headquarters in Paderborn, Germany, supporting clients in the gas, water and electric industries. We are dedicated to enabling the owners and operators of utility assets to plan and manage their networks more successfully.

Mettenmeier GmbH provides a SIEMENS certified interface between the Smallworld standard Network Resource Managers and PSS SINCAL.

Mettenmeier GmbH Utility Solutions
Mr Benjamin Pehle
Klingenderstr. 10 – 14
33100 Paderborn
Germany
Fon +49 5251 150-375
Fax +49 5251 150-366
e-mail benjamin.pehle@mettenmeier.de
http://mettenmeier.de

Khatib & Alami – C.E.C.

Khatib & Alami – Consolidated Engineering Company (K&A) is a multi-disciplinary company specialized in consulting engineering studies, design and construction supervision of architectural, structural, civil, electrical, mechanical, industrial, environmental, transportation, telecommunication, information technology and geographic information systems (GIS) projects.

K&A provides an interface between ArcGIS and PSS SINCAL. This interface is for electricity transmission and distribution networks.

Khatib & Alami – C.E.C.
GIS Services Division
Mr Bilal A. Hassan
P.O. Box 2732, Abu Dhabi – UAE
Fon +971-2-6767300 X202
Fax +971-2-6767070
e-mail bahassan@gisadwea.ae
http://www.khatibalami.com